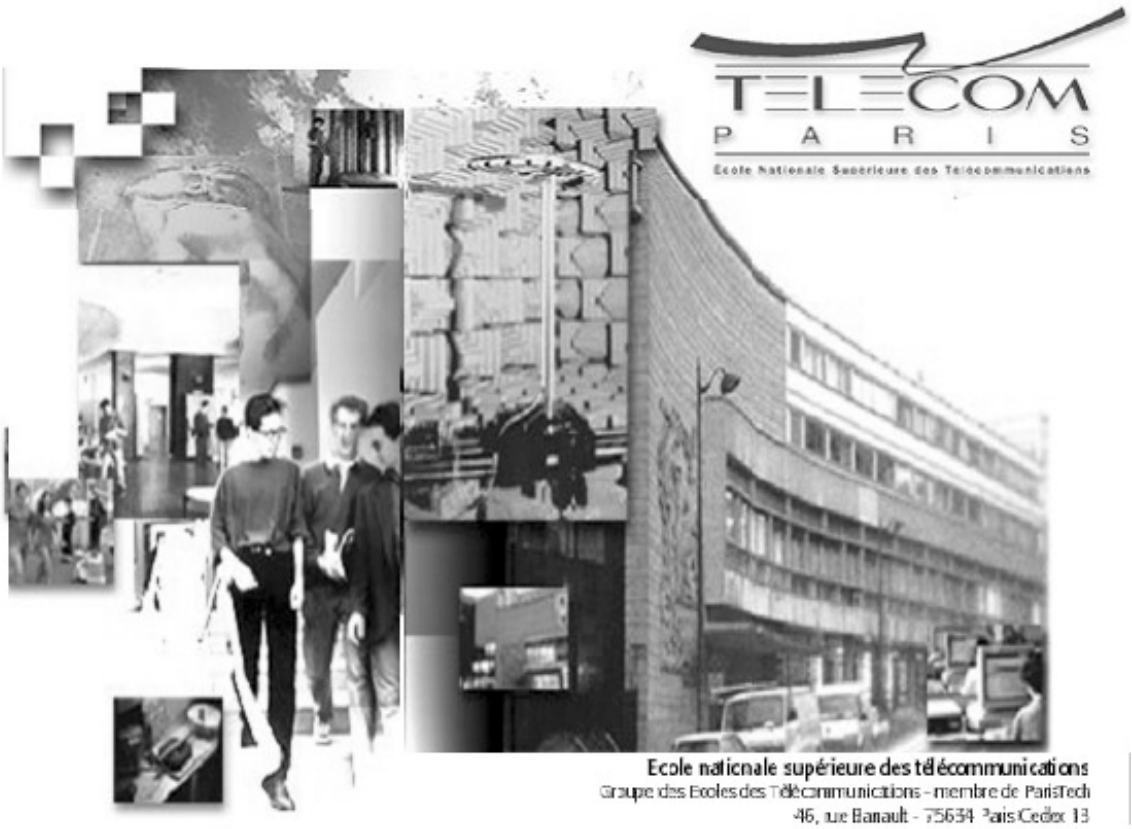

TELECOM
PARIS
Ecole nationale supérieure des télécommunications
Ecole nationale supérieure des télécommunications
Groupe des Ecoles des Télécommunications - membre de ParisTech
46, rue Barrault - 75634 Paris Cedex 13

SOMMAIRE

Présentation de L'École *3*

Les missions : La Recherche **3**
Quelques Chiffres de la Recherche 4
Les Départements de la Recherche 4

Le Département TSI (Traitement du signal et de l'image) **4**
Axes de recherche 5
Les missions du département 5
L'organisation du département TSI 5

Le Groupe PAM (Perception, Apprentissage et Modélisation) **6**
Présentation de l'équipe 6
Axes de Recherche 7
Les chercheurs du Groupe PAM 7

Présentation du Travail *8*

Présentation du Projet MAJORDOME **8**
Qu'est-ce qu'un Majordome ? 9
Systèmes classiques de messagerie et Majordome 9
Avantages et Potentialités du Majordome 10

Organisation du Projet **10**
Découpage du Projet en sous projets 10
Sous-Projet 1 10

Présentation de mon Travail **11**

Commentaires et Évaluation *12*

Présentation du matériel **12**
Environnements utilisés 12
Logiciels utilisés 12

Journal de Bord Hebdomadaire **13**

Travail effectué **14**
Description de l'évaluation d'un système de résumé automatique 14
Description des données linguistiques 18
Brève description de l'analyse 19
Astuces, Problèmes et Évolution du logiciel 23

Connaissances acquises **29**

Remerciements **30**

Bibliographie et Annexes Techniques *31*

Bibliographie **31**

Webographie **31**

Annexes Techniques **32**

Présentation de L'École

En développant constamment les échanges avec des partenaires exigeants du monde de la Recherche et de l'Entreprise, l'École dispense aujourd'hui un enseignement qui la situe au cœur de la Société de l'Information.

Elle accueille plus de 1 000 étudiants, toutes formations confondues et plus de 140 enseignants-chercheurs. Elle a essaimé à Toulouse et à Sophia Antipolis.

Depuis le 1er janvier 1997 elle appartient au **Groupe des Ecoles des Télécommunications**, établissement public administratif qui comporte aussi ***l'Ecole nationale des télécommunications de Bretagne*** et ***l'Institut national des télécommunications*** d'Evry, et contribue à constituer un pôle de tout premier plan pour l'enseignement et la recherche dans le vaste domaine des sciences de l'information et de la communication.

Télécom Paris est également membre de **ParisTech** (Paris Institut of Technology) qui rassemble les huit autres écoles suivantes : **Agro, Arts et Métiers, Chimie de Paris, Eaux et Forêts, Génie Rural, Mines, Physique chimie de Paris, Ponts et Techniques Avancées.**

Les missions : La Recherche

La recherche a un triple objectif : garantir un contenu pédagogique de haute qualité, produire des savoirs et mener des actions de recherche appliquée.
L'organisation de l'École ***en départements d'enseignement et de recherche***, où chaque option d'enseignement s'appuyant sur une équipe de chercheurs, maintient une étroite imbrication entre ces activités.

La recherche est orientée autour des pôles majeurs que sont **les communications, le traitement des signaux et des images, et l'économie des systèmes d'information.**

L'École a déjà acquis une position de pointe, reconnue au plan international, dans plusieurs domaines : **traitement du signal, traitement de la parole et de l'image, circuits intégrés associés.** Elle développe actuellement de nouveaux pôles d'excellence dans ***les architectures parallèles, le traitement automatique de la parole, les spécifications formelles et les réseaux à haut débit.***

La recherche s'effectue principalement avec les universités et les grands organismes de recherche, notamment le CNRS. La recherche appliquée se développe par les relations contractuelles nouées entre l'École et les industriels ainsi que **France Télécom R&D**, avec lequel l'ENST entretient de fortes et anciennes relations.

Les départements travaillent tous en forte relation avec les industriels et les plus grands centres de recherche français et étrangers.

Quelques Chiffres de la Recherche

- ✓ 125 enseignants-chercheurs,
- ✓ 15 chercheurs CNRS,
- ✓ plus de 50 doctorats par an.
- ✓ plus de **400** publications

- ✓ 4 départements d'enseignement et de recherche

Les Départements de la Recherche

- ➢ COMELEC (Communications et électronique)
- ➢ EGSH (Economie, gestion, sciences sociales et humaines)
- ➢ INFRES (Informatique et réseaux)
- ➢ TSI (Traitement du signal et images)

et **une unité de recherche associée au CNRS**, l'URA 820 *(Traitement et communication de l'information).*

Le Département TSI (Traitement du signal et de l'image)

Le traitement des signaux a récemment connu un développement important dans le domaine théorique et dans ses applications aux télécommunications.

La force de Télécom Paris est d'être active sur les outils théoriques : séparation de source, traitements statistique, analyse temps-fréquence... et sur les applications dans des contextes extrêmement variés : production de sons pour les instruments de musique et la parole, restauration d'enregistrements sonores, réseaux d'antennes acoustiques, annulation d'écho, égalisation de canaux de transmission, reconnaissance de parole, identification de locuteur, compression de source, interfaces.

Dans le domaine de l'image, la mission principale est de concevoir et de mettre en oeuvre les actions d'enseignement et de recherche des méthodes de traitement de l'information visuelle et

des techniques de sa représentation. L'activité est organisée autour des trois formes complémentaires de l'image ; l'image de télédiffusion et en particulier pour la télévision numérique, le traitement numérique de l'image et son impact sur le monde moderne, enfin l'image optique traitée par des méthodes analogiques ou hybrides.

Axes de recherche

Traitement statistique du signal
Traitements multicapteurs
Traitement de la parole et du son
Traitement du signal pour les communications
Cartographie satellite radar ou optique
Imagerie cérébrale
Perception et traitement de la couleur
Traitement des objets 3D
Interaction homme-machine
Caractérisation optique des matériaux de stockage

Les missions du département

- Le département TSI a pour missions ***l'enseignement*** (initial et continu), ***la recherche*** (académique et contractuelle), et ***la formation par la recherche*** dans les domaines du traitement du signal et des images et de l'application du traitement du signal et des images dans divers contextes de la société de l'information dont les télécommunications.

- La recherche méthodologique et fondamentale, en relation étroite avec les organismes nationaux et internationaux de coordination de la recherche et en particulier le **CNRS** ; elle permet au département de contribuer à l'innovation par la découverte de concepts nouveaux ; La recherche appliquée, souvent menée ***en collaboration avec des partenaires industriels français ou étrangers***, elle garantit un contact permanent avec les technologies émergentes ainsi qu'avec les nouveaux usages.

- Le département TSI participe au rayonnement de l'École en la représentant, dans son domaine d'activité, auprès des différentes instances et des organismes nationaux ou internationaux (CNRS, IEEE, RNRT, etc…) et en participant de façon active à la vie scientifique nationale.

L'organisation du département TSI

Le département TSI est organisé en 5 groupes :

Groupe «Traitement et Interprétation des Images» (TII)

Ce groupe conduit des recherches sur la mise en oeuvre de schémas complets de traitement, d'analyse et d'interprétation d'images, en particulier de scènes complexes.

Groupe «Traitements Statistiques et Applications aux Communications» (TSAC)

Ce groupe travaille dans *le signal pour les communications* ; *la séparation de sources* ; *la modélisation statistique pour le signal et l'image* (la reconstruction et la restauration d'images, le télétraffic (analyse et modélisation)).

Groupe «Perception, Apprentissage et Modélisation» (PAM)

Ce groupe étudie le rôle des facteurs humains dans l'accès aux divers types d'information :

- **La parole** : reconnaissance et identification de locuteurs ;
- **L'image** : psychovision (perception du contraste, de la couleur, du relief), et imagerie de très haute qualité ;
- **L'écrit** : fax, structuration des documents ;
- **La fusion des modalités perceptives dans l'appréhension de l'environnement** ;
- **Les interfaces multimodales.**

Groupe «Codage» (COD)

Ce groupe travaille sur des techniques éprouvées *de compression de sources* ainsi que sur leur adaptation aux applications de l'audiovisuel et du multimédia.

Groupe «Audio, Acoustique et Ondes» (AAO)

Ce groupe étudie *la physique des ondes* dans les deux domaines de l'optique et de l'acoustique.

Le Groupe PAM (Perception, Apprentissage et Modélisation)

Responsable : Hans Brettel

Présentation de l'équipe

L'équipe ***Perception, Apprentissage, Modélisation (PAM)*** s'est formée ***dans le département Traitement du Signal et des Images (TSI)***, afin de réunir les chercheurs dont les études touchent aux *sciences cognitives* aussi bien qu'au *traitement des signaux* ou à *l'analyse des images*. Autant que les techniques, ce sont les approches qui se sont révélées complémentaires : les chercheurs rassemblés dans cette équipe étudient la perception ou le contrôle sensori-moteur.

Elle se présente donc en tant *qu'**une des équipes de l'Opération de Recherche Traitement du Signal et des Images de l'URA 820***, cela d'autant plus naturellement qu'une majorité de ses membres sont chercheurs CNRS.

La formation du groupe **Perception, Apprentissage et Modélisation**, dont ***les thèmes de recherche vont de la stimulation sensorielle à la cognition***, permet donc de faire interagir des chercheurs réunis autour de thèmes communs :

- ✓ la fusion des informations,
- ✓ l'apprentissage,
- ✓ la variabilité interindividuelle
- ✓ l'organisation du geste.

Axes de Recherche

- Modélisation de la perception des couleurs
- Etude de la variabilité de l'écriture
- Documents multimédia
- Fusion de données pour l'identification des personnes
- Modélisation de la production et de la perception de la parole
- Modélisation du contrôle moteur et de la fusion des informations sensorielles

Documents multimédia

Claudie Faure, Laurence Likforman, en collaboration avec G. Chollet

Comprendre un document nécessite d'en reconnaître les structures physiques et logiques. La première étape de reconnaissance de la structure physique fait apparaître les blocs de texte, les lignes, les mots. Au cours de la seconde étape de reconnaissance de la structure logique, les entités physiques sont interprétées comme des entités signifiantes pour la communication écrite.

Traitement et accès aux documents multimédia

Nos travaux sur la structuration physique des documents manuscrits nous amènent à étudier la structure logique de ces documents tout-venant que sont **les télécopies**. Ces documents sont généralement imprimés ou mixtes (contenant des parties imprimées et manuscrites). La reconnaissance de la structure logique ne peut s'appuyer ici sur un modèle a priori de document, précisé à l'avance dans une feuille de style. Nous cherchons donc à dégager un modèle générique de présentation physique, s'appuyant sur les **conventions de la communication écrite**, en nous aidant de **considérations linguistiques**, notamment **la reconnaissance de mots-clés.**

Des partenaires industriels seront recherchés pour cette étude. ***Le but est d'offrir de nouveaux services de télécommunications permettant d'accéder au contenu des images de télécopies, en identifiant des champs particuliers : nom du ou des destinataires, de l'envoyeur, ses coordonnées, la date, l'objet du message.***

Les chercheurs du Groupe PAM

Permanents
Gérard CHOLLET
Laurence LIKFORMAN
Claudie FAURE
François YVON

Stage Long
Pascal VAILLANT

Présentation du Travail

Présentation du Projet MAJORDOME

Le projet Majordome est l'élaboration d'**un système de messagerie unifiée** interrogeable à partir d'un téléphone mobile. Majordome est aussi un assistant électronique personnel.

Les partenaires Européens de ce projet sont :

Software 602 (Tchéquie)
Entreprise de solutions de communications : messageries, serveurs de bases de données

Airtel (Espagne)
Grande entreprise de télécommunications (équipements, réseaux, communication mobile)

UPC (Université Polytechnique de Catalogne, Espagne)
Utilisation de ressources lexicales multilingues, extraction d'information dans les documents, implémentation d'interfaces en langage naturel, représentation des connaissances…

KTH (Royal Institute of Technology, Suede)
Le Département d'Analyse Numérique et d'Informatique mène des activités de recherche en réseaux de neurones, vision, interaction homme-machine, traitement du langage naturel.
En particulier, le NADA a développé des outils pour le résumé automatique de textes et l'extraction de mots clés pour l'Anglais et le Suédois.

Euroseek (Suède)
Entreprise de services électroniques : commerce électroniques, portails

GET-ENST (Ecole Nationale Supérieure des Télécommunications, France)
Reconnaissance de la parole et sa synthèse, analyse documents et le traitement des images

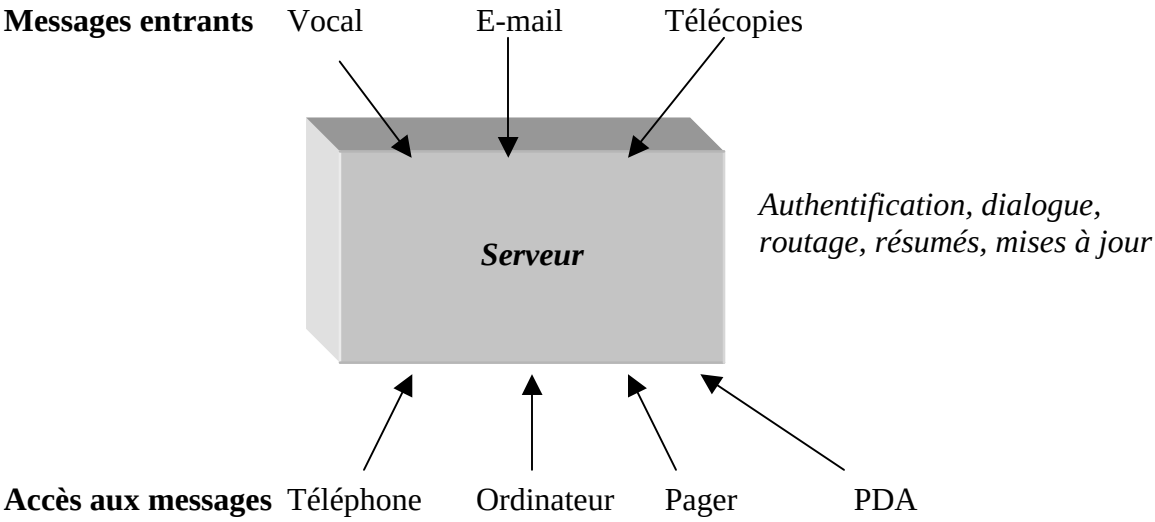
EDF-DER (France)

HOLISTIQUE Communication

Le projet MAJORDOME est basé sur un système de messagerie unifiée (système qui permet d'accéder aux messages vocaux, mails et télécopies dans une unique boîte aux lettres).

Le contenu des messages sera en grande partie interprété et présenté de manière condensée. Ceci permettra d'accéder directement ou à distance via Internet ou un téléphone mobile.

Qu'est-ce qu'un Majordome ?



Systèmes classiques de messagerie et Majordome

	Systèmes de messagerie	**MAJORDOME**
messages vocaux	N° de téléphone émetteur Date	+ nom de l'émetteur + nom du destinataire + identification du locuteur
images de textes	N° du télécopieur Date	+ nom et coordonnées de l'émetteur + nom et coordonnées du destinataire + objet du message ***+ extraction de mots- clés***
e-mail	Adresse mail de l'émetteur Date Objet du message	***+ extraction de mots- clés*** ***+ résumé de messages***
accès à la messagerie	Code DTMF	+ mot de passe vocal + vérification du locuteur + synthèse vocale des messages

Dans MAJORDOME, l'innovation consiste en l'accès au contenu des messages. Les messages seront interprétés pour en extraire ***les informations synthétiques*** : noms et coordonnées de l'expéditeur et du destinataire, date, objet du message, résumé du message (pour les mails et les pages web). Ces informations sont ensuite restituées vocalement par synthèse. Pour les messages reçus sous forme d'image, une reconnaissance de caractères est nécessaire.

Les technologies mises en œuvre sont celles :

- Du traitement de la parole
- Du traitement de l'écrit
- Traitement du langage naturel (TALN) ***(extraction de mots clés, analyse des contenus, résumé de messages)***
- La communication Homme/Machine

Avantages et Potentialités du Majordome

Son utilisation accroîtrait les capacités de communication des entreprises, en Internet et en externe, et rendrait plus aisées le travail à distance de ses agents.

Elle faciliterait, en outre, la création de réseaux d'entreprises qui souhaiteraient mettre en commun leurs savoir-faire.

La réception d'Internet sur les téléphones mobiles constitue un des grands axes de recherche de développement des opérateurs de télécommunications, lesquels pourraient d'ailleurs, par la suite, proposer de nouveaux services complémentaires aux utilisateurs.

Organisation du Projet

Software602 développe le système de messagerie unifiée. La reconnaissance vocale et l'analyse des images de télécopies sera développée par l'URA CNRS 820 à l'ENST
La partie traitement du langage naturel sera traitée par l'UPC et le KTH.

Découpage du Projet en sous projets

Chef de Projet – France : HOLISTIQUE Communication

Sous-projet 1 : Extraction d'information dans les images de textes
Sous-projet 2 : Traitement de la Parole : vérification et identification du locuteur
Sous-projet 3 : Traitement de la Parole : reconnaissance des noms propres
Sous-projet 4 : Dialogue Vocal
Sous-projet 5 : Évaluation

Les messages arrivant dans une messagerie sous formes diverses : mail, télécopies, documents attachés, messages vocaux. La reconnaissance et l'extraction d'informations ciblées dans les images permet une restitution vocale notamment.

Sous-Projet 1

Description

Responsable : Laurence LIKFORMAN (GET-ENST)
Partenaires : GET-ENST (France), UPC (Espagne), KTH (Suède)

Ce sous-projet a pour but ***l'extraction automatique d'informations ciblées dans les images de texte*** : télécopies, documents numérisés attachés aux mails. Ces informations concernent les champs d'identification de l'émetteur et du destinataire *(noms, prénoms, coordonnées –téléphone, fax, adresses-, date, objet du message)* ainsi que des **mots clés ou un résumé relatifs au corps du message**. En effet, l'identité de l'émetteur donne des indications sur le contenu.

Le résumé automatique du message est envisagé pour les documents de bonne qualité (documents scannés à haute résolution, télécopies électroniques), après extraction du corps du message.

La réalisation de ces fonctionnalités fait appel aux techniques suivantes :

- analyse de documents ;
- reconnaissance de l'écriture ;
- **traitement du langage *(constitution de résumés, filtrage des informations extraites).***

Objectifs

Développer un **outil informatique d'extraction d'information dans les images de textes**. Cet outil peut se décomposer en :

- construction d'une base de données d'images de textes
- développement d'une méthode de reconnaissance des caractères dégradés
- développement d'une méthode de segmentation des images de texte
- développement d'une méthode d'extraction d'information pour l'étiquetage des différents champs d'identification et ***l'extraction de mots clés dans le corps du message***

Présentation de mon Travail

L'objectif de ce stage qui s'est déroulé du 2 juillet au 28 septembre 2001, était une contribution au ***traitement et à l'interprétation des messages électroniques afin de résumer chaque message en une ou deux phrases.*** Ce travail s'inscrivait dans le cadre du projet Majordome, système de messagerie unifiée interrogeable à partir d'un téléphone mobile.

Dans un premier temps, il s'est agi ***d'étudier les standards de messageries et d'évaluer les logiciels de résumé de texte*** de Messieurs Hercules DALIANIS & Martin HASSEL, partenaires Suédois du projet Majordome.

Dans un deuxième temps, à l'aide du dictionnaire du LADL (dictionnaire ABU), dictionnaire lemmatisé, il s'agissait ***de générer une ou deux phrases qui résume un mail automatique.***

Les manipulations effectuées devraient permettre à la personne recevant un grand nombre de messages, d'entretenir un dialogue utile avec le standard de messagerie. Désormais, grâce au logiciel, ***la personne, pourra par exemple prendre connaissance des***

messages provenant de tel ou tel correspondant ou bien savoir quels sont les mails concernant tel ou tel sujet attendu, demander le détail d'un message résumé ou la liste des messages en réponse à une demande.

Commentaires et Évaluation

Présentation du matériel

Tous les calculateurs, stations et terminaux graphiques communiquent entre eux et avec les autres machines de l'École par le réseau Ethernet. Les postes de travail sur lesquels on peut se connecter sont soit **des stations SUN**, soit **des terminaux X**, soit **les postes de travail du centre de calcul.**

Environnements utilisés

- ✓ Unix sous une station Sun Solaris Ultra 5
- ✓ Linux
- ✓ X-windows
- ✓ Windows NT 4

Logiciels utilisés

Sous Unix :

- PINE (programme pour messagerie électronique, conforme à la norme MIME)
- Mail tool version 3.6
- Résumidor0 (logiciel de résumé automatique espagnol, crée par des gens de l'UPC Universtat Politecnica de Catalunya, et fait sous Perl)
- Relax-3.4 (logiciel de désambiguïsation morpho-syntaxique pour l'espagnol)
- Xemacs
- Textedit

Sous Windows :

- Visual C++ 6
- Résumé Automatique du KTH (accessible par le Web)

Manipulations :

- Telnet
- Connexion d'un PC à une station Sun Solaris par ftp
- Transferts de fichiers d'un PC à une station Sun Solaris par ftp

- Transferts de fichiers d'une station Sun Solaris à un PC par ftp
- Imprimer sous une station Sun
- Dézipper un fichier sous Unix par la commande gunzip
- Copier un fichier d'une Sun à un PC par ftp
- Copier un fichier d'un PC à une Sun par ftp
- Gestion d'un compte unix par ftp sous windows NT
- Conversion d'un fichier texte dos en fichier texte unix par la commande dos2unix
- Conversion d'un fichier texte unix en fichier texte dos par la commande unix2dos

Journal de Bord Hebdomadaire

	JUILLET	**AOUT**	**SEPTEMBRE**
Semaine 1	- Mise en place - Connaissance du Projet MAJORDOME - Lectures & Recherches bibliographiques - Familiarisation & Apprentissages du monde Unix - Premières idées du logiciel de résumé automatique *(recherches linguistiques et début formalisation)*	- Début code fonction 2 - Recherche de mots-clefs de la fonction 2 fonctionne - Suite évaluation du logiciel de résumé automatique suédois - Familiarisation & Apprentissages du monde Unix - Optimisation de la fonction 1	- Fin évaluation du logiciel suédois - Rédaction d'un petit article concernant l'évaluation du logiciel de résumé automatique suédois dans le rapport d'activité du projet MAJORDOME - Sélection des phrases de la fonction 3 fonctionne
Semaine 2	- Début de l'évaluation du logiciel Suédois avec statistiques manuelles sur les textes résumés - Apprentissage de l'évaluation des systèmes de résumés automatiques - Familiarisation & Apprentissages du monde Unix - Analyse de mon logiciel de résumé automatique - Étude du Résumidor0 - Simulation manuelle des fonctions 1 & 2 de mon analyse *(modifications apportées à la fonction 2)* - Élaboration des premiers algorithmes pour les fonctions 1 & 2	- Suite évaluation du logiciel de résumé automatique suédois - Suite et modification du code de la fonction 2 (ajout de fonctions) - Tests de mon logiciel sous Unix (modification sur les « include » et quelques fonctions prédéfinies) - Fonction 2 de mon logiciel fonctionne	- Optimisation du logiciel (apport linguistique grâce à une fonction gérant les connecteurs logiques) - Adaptation de mon logiciel sous Unix - Rédaction du Rapport de Stage en vue d'expliquer l'analyse linguistique de mon logiciel
Semaine 3	- Évaluation du logiciel de résumé automatique suédois - Élaboration des premiers algorithmes pour la fonction 3 - Écriture du « .h » - Début du code de la fonction 1	- Suite évaluation du logiciel de résumé automatique suédois - Début du code de la fonction 3 - Modification du « .h » - Recherche dans le dictionnaire à optimiser fortement si le temps le permet	- Suite de l'adaptation de mon logiciel sous Unix - Amélioration de l'interface - Rédaction d'un rapport d'activité sur l'Évaluation du logiciel de résumé automatique suédois

Semaine 4	- Suite évaluation du logiciel de résumé automatique suédois - Participation à la réunion des partenaires de MAJORDOME, chez EDF - Élaboration de l'algorithme de la fonction 1 - Modification du « .h » - Modification du code de la fonction 1 - Suppression des mots parasites du dictionnaire afin d'avoir un dictionnaire ne contenant que des mots-clés - Modification du dictionnaire de mots-clés - Fonction 1 de mon logiciel fonctionne	- Modification du code de la fonction 3 - Modification du « .h » - Suite évaluation du logiciel de résumé automatique suédois - Fonction 3 de mon logiciel fonctionne - Logiciel de base fonctionne sous Windows	- Logiciel de résumé automatique de textes fonctionne sous Unix & Windows - Rédaction d'un mode d'emploi pour mon logiciel - Rédaction du Rapport de Stage en vue d'expliquer l'analyse informatique de mon logiciel - Rédaction d'un rapport d'activité sur l'Évaluation de mon logiciel avec celui du logiciel de résumé automatique suédois
Semaine 5		- Modification du code de la fonction 3 - Conclusions sur l'évaluation du logiciel suédois - Rédaction d'un rapport d'activité concernant l'évaluation du logiciel suédois	

Travail effectué

J'ai élaboré un logiciel de résumé automatique de textes, basé sur la ***fréquence d'occurrences de mots-clés rencontrés dans le texte source.***

Description de l'évaluation d'un système de résumé automatique

Étant donné que l'ENST n'avait pas la possibilité de mobiliser un grand nombre de sujets experts pour faire des tests extensifs des systèmes de résumé, comme ils le font dans la campagne **d'évaluation TIPSTER/SUMMAC**, j'ai dû le faire moi-même en me mettant dans la peau d'un sujet expert…

Voici une fiche type de chaque texte évalué avec le logiciel suédois :

Fiche numero :

Nombre de mots du texte source :
Nombre de mots du texte cible :
Contraction du resume : %
Type de texte : journal ou académique
Mot(s) cle(s) :

Informations

Qui : oui/moyen/non/pas indique dans le texte source
Quoi : oui/moyen/non/pas indique dans le texte source
Ou : oui/moyen/non/pas indique dans le texte source
Comment : oui/moyen/non/pas indique dans le texte source
Pourquoi : oui/moyen/non/pas indique dans le texte source

Contexte : OK (texte source = texte cible), partiellement OK, informations oubliées
Longueur : bonne/moyenne/trop court…
Intelligibilite : bonne/moyenne/peu claire

Remarque(s) :

Informations :
Parfois, les réponses aux questions primaires n'étaient pas présentes dans le texte source. Dans ce cas, je l'indiquais entre parenthèses ou directement.

Contexte :
Si le contexte du résumé reflétait point à point l'idée maîtresse du texte source alors je l'indiquais en notant : *OK.*
Si le résumé comportait les idées générales du texte source sans les expliquer alors je notais : *partiellement OK*.
Si le logiciel n'avait pas su redonner toutes les idées dans le résumé alors je notais : *informations oubliées*.

Longueur :
Je prenais comme critère le taux de contraction du résumé produit par rapport au texte source.
Si la contraction était autour des 30%, je notais : *bonne*.
Si la contraction était moins de 20%, je notais : *un peu trop court*.
Si la contraction était autour de 40%, je notais : *moyenne*.
Si la contraction était de plus de 50%, je notais : *un peu trop long.*

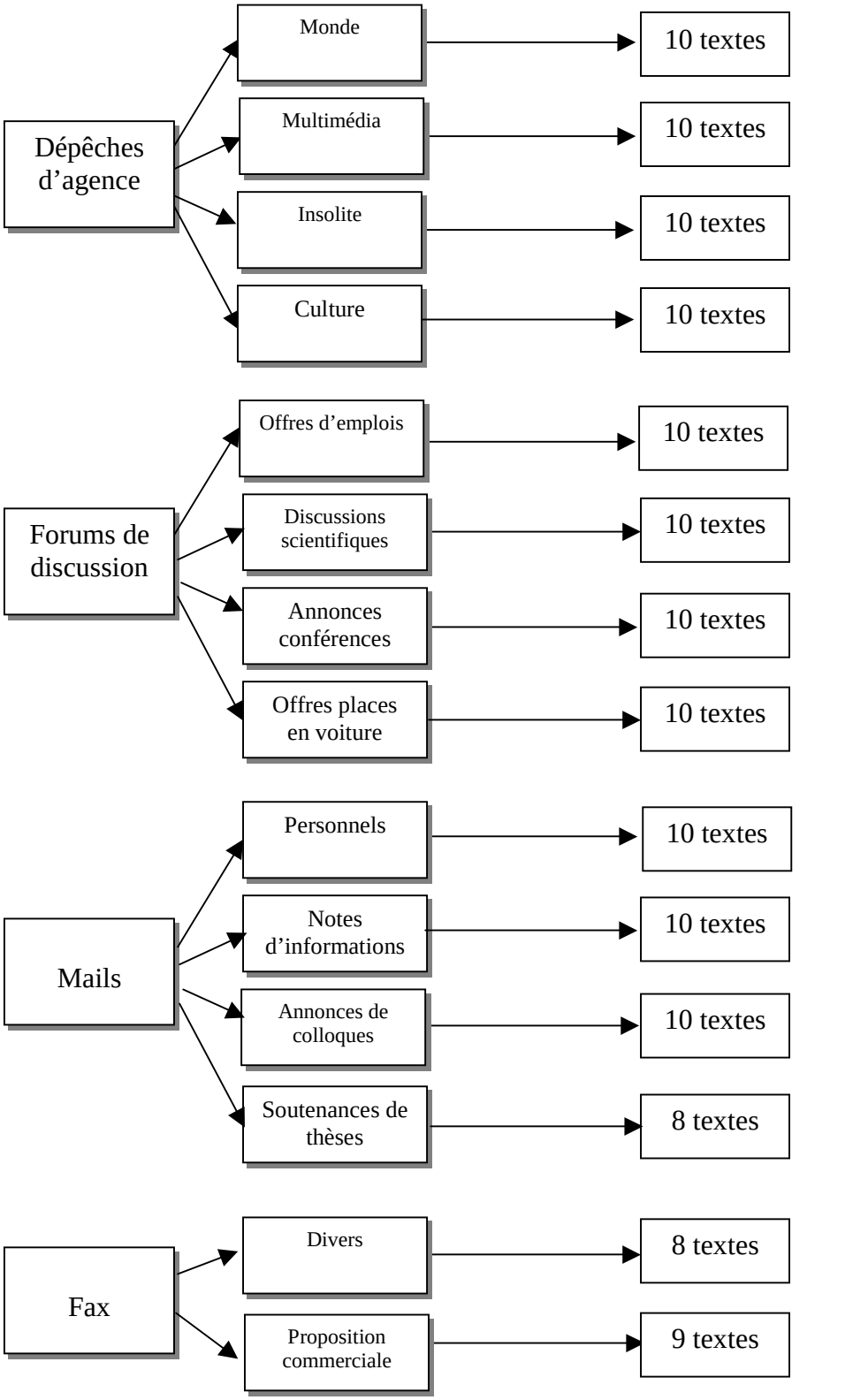
Intelligibilité :
J'observais la cohérence du résumé produit, en remarquant par exemple si des phrases contradictoires étaient collées côte à côte ou si lors de la lecture, la logique de pensée du résumé était fluide.

Remarque(s) :
Je notais ici, ce que je ne pouvais écrire ailleurs : les interrogations ou réflexions que j'avais pu tirer de la production d'un résumé.

Les conditions dans lesquelles je travaillais étaient les suivantes :

- Taux de contraction pour le résumé : 30%
- Textes français à résumer en français
- Résumé de style journalistique plutôt académique
- Affichage des mots-clefs reconnus dans le texte source

J'ai procédé à ***135 fiches d'évaluation***. En effet, j'ai opté pour différents sujets et sous catégories :


Dépêches d'agence
Monde
Multimédia
Insolite
Culture
10 textes
10 textes
10 textes
10 textes
Forums de discussion
Offres d'emplois
Discussions scientifiques
Annonces conférences
Offres places en voiture
10 textes
10 textes
10 textes
10 textes
Mails
Personnels
Notes d'informations
Annonces de colloques
Soutenances de thèses
10 textes
10 textes
10 textes
8 textes
Fax
Divers
Proposition commerciale
8 textes
9 textes
1. Catégories Générales
2. Catégories Spécifiques
3. Évaluation des textes

CONCLUSION de l'Évaluation

Les résultats ont divergé aussi bien selon la longueur des textes que leur contenu.

En ce qui concerne **les mails**, ***le logiciel ne supportait pas les ponctuations stylistiques*** que l'on peut rencontrer dans les mails (par exemple, des tirets mis côte à côte formant un trait de soulignement). En effet, dans ce cas, le logiciel ne donnait aucun résumé ou plutôt il donnait un message d'erreur : « Trop de données ».

En ce qui concerne **les mails personnels ou les dépêches d'agence**, quand il y avait des ***phrases trop longues***, le logiciel ne pouvait calculer une bonne contraction de texte. En effet, le logiciel pouvait produire des résumés comportant deux phrases et dont la contraction était de 50 % !

En ce qui concerne **les forums de discussion**, le logiciel ne prenait pas en compte des éléments pertinents tels que ***des horaires, des dates, des lieux, des noms*** lorsqu'ils étaient seuls (et non insérés à l'intérieur d'une phrase). En effet, les résumés produits ne redonnaient plus que le corps du message, sans trace de l'auteur et des informations concernant son annonce de conférence... Ce désagrément touchait aussi les mails qui annonçaient des colloques ou des séminaires.

En ce qui concerne **les fax**, il a fallu tout d'abord, les trier puisque la plupart étaient illisibles. Puis, après évaluation, je me suis rendu-compte qu'il restait ***peu de phrases ou mots interprétables***, ce qui entraînait ***un résumé souvent incohérent*** au niveau de la syntaxe et au niveau sémantique. En effet, le logiciel repérait ***peu ou aucun mot-clé*** et ne redonnait pas toujours les informations relatives au nom du destinataire, de l'expéditeur, à la date, à l'objet lui-même... Parfois, même, le logiciel donnait une contraction de résumé de 100% alors que le nombre de mos du texte source était faux ou que des informations issues du texte source étaient inexistantes dans le texte cible.

D'une manière générale, le logiciel ne peut prétendre à reconnaître deux phrases différentes dans le cas ou un point de fin de phrase (ou tout autre caractère de fin de phrase) aurait été oublié entre les deux phrases. Le calcul de mots-clés pourrait alors relever cette phrase comme étant une phrase importante du texte parce qu'elle contient plusieurs mots-clés.

Description des données linguistiques

J'avais à ma disposition, ***un dictionnaire lemmatisé*** (ABU) qu'il a fallu remanier de façon à pouvoir extraire, au moment de la recherche dans le dictionnaire, uniquement des **mots clés.**

Le dictionnaire a été débarassé des :

- Pronoms : personnels, compléments, réfléchis…
- Conjonctions

- Adverbes les plus génériques
- Interjections
- Prépositions
- formes conjuguées des verbes « avoir », « être », et « faire »
- d'autres mots dits « stopwords »

Brève description de l'analyse

L'analyse de mon résumé automatique comporte trois fonctions principales :

- Découpage du texte source
  - Deux phases :
    - Gestion d'ouverture en lecture du fichier source
    - Prédécoupage du texte
  - Buts :
    - Différencier un mot d'une ponctuation
    - Numéroter les phrases et les paragraphes éventuels
- Calcul des Fréquence des mots clés grâce au dictionnaire
  - Deux phases :
    - Recherche des mots clés
    - Calcul des occurrences des mots clés
  - Buts :
    - Savoir si le mot appartient au dictionnaire des mots-clés
    - Compter les occurrences desdits mots-clés
- Sélection des phrases
  - Trois phases :
    - Détermination du score des phrases
    - Sélection par le biais d'un flag, des phrases les plus fortes
    - Gestion d'écriture dans le fichier destination
  - Buts :
    - Calculer le score d'1 phrase par rapport au nombre de mots-clés qu'elle contient
    - Ne prendre en compte pour la réécriture dans le fichier cible (le résumé) que les phrases les plus fortes

Les Règles d'or

1. Tout texte doit être stocké en mémoire sous forme d'une **liste chaînée**

Définition de la structure ELEMENT :

Mot (mot ou ponctuation)	*char**
Pointeur vers suivant	*struc element*
Pointeur vers precedent	*struc element*

Nombre d'occurrences des mots-clés dans le texte	*int*
Numéro de phrase	*int*
Numéro de paragraphe	*int*
Mot-clé	*char (0 =F ; 1=T)*

```
/* cette structure contiendra les mots et leurs attributs */

typedef struct Element
 {
 unsigned char Mot[255];          /* contient le mot ou une ponctuation */
                                  /* 255 pour les tests voir avec malloc apres */
 struct Element *ElementSuivant;  /* contient un pointeur vers le mot suivant */
 struct Element *ElementPrecedent;/* contient un pointeur vers le mot precedent */
 int nbOcc;                       /* contient le nombre d'occurrences du mot */
 int nbMotsPhrase;                /* contient le nombre de mots dans une phrase*/
 int noPhrase;                    /* contient le numero de la phrase a laquelle
                                  appartient le mot */
 int noParagraphe;                /* contient le numero de paragraphe auquel
                                  appartient le mot */
 char MotClef;                    /* contient un mot cle T(true) sinon F(false) */
}tabElement;
```

2. Si element est de la ponctuation, alors, le **nombre d'occurrences est de 0**.
3. Si il y a plus d'un retour à la ligne, alors **on incrémente le numéro de paragraphe**.

4. Dans le module 2, on va tout d'abord, rechercher les mots-clés, puis on va **compter le nombre d'occurrences de chacun des mots-clés.**

Définition des structures TOCC & TABOCC :

```
/* cette structure contiendra les occurrences des mots cles */

typedef struct TOCC
 {
 unsigned char Mot[26];   /* contient le mot clef et ses attributs */
 int nbOcc;               /* contient le nombre d'occurrences du mot-clef */
 }OCC;

typedef struct TABOCC
{
OCC * tabOcc;             /* tableau d'occurrences */
 int nbMotsCles;          /* nombre de mots-cles trouves */
 }TAB_OCC;
```

5. Dans le module 3, on va faire **un tableau de score des phrases.**

Définition de la structure TSCORE :

```
/* cette structure contiendra les scores des phrases */

typedef struct TSCORE
{
 int noPhrase;          /* contient le numero de la phrase */
 int ScoreMotsCles;     /* contient le score des mots-cles de la phrase */
 int nbMots;            /* contient le nombre de mots de la phrase */
 char Flag;             /* contient un booleen T(true) ou F(false)indiquant
                           si la phrase est a selectionner */
 }SCORE;
```

6. **Variables Globales** :
   - nombre de mots total
   - nombre de phrases total
   - nombre de paragraphes total
   - nombre entré par l'utilisateur pour la contraction de résume souhaitée

Algorithmes

- Découpage du texte source

Allocation du premier mot
Allocation du premier vrai mot
Par défaut, on est en Mode ponctuation

Tant qu'on est pas à la fin du fichier source
    Lecture du caractère courant
        Si le caractère courant est une ponctuation
            Si on est en Mode mot, alors on a trouvé un mot
                On met 1 dans la case nbOcc de ce mot
                Initialisation par défaut du maillon
                Incrémentation du nombre de mots au total
                On passe en Mode ponctuation
            On passe au caractère suivant
        Sinon
            Si on est en Mode mot, alors on a trouvé une ponctuation
                On met 0 dans la case nbOcc de ce mot
                Initialisation par défaut du maillon
                On met 0 dans la case MotClef de la ponctuation
                Si on détecte une ponctuation de fin de phrase
                    Incrémentation du nombre de phrases au total
                    Mise à jour du numéro de phrase du futur maillon
                Si on détecte une fin de paragraphe
                    Incrémentation du nombre de paragraphes au total
                    Mise à jour du numéro de paragraphe du futur maillon
                On passe en Mode mot
            On passe au caractère suivant

On met un caractère de fin de chaîne à la position du caractère dans le mot
Fermeture du fichier source

- Calcul des Fréquence des mots clés grâce au dictionnaire

Pour chaque mot (mot ou ponctuation) du texte source
    Si le mot courant appartient au dictionnaire des mots-clefs
        On met 1 dans la case MotClef de ce mot
        Si le mot a déjà été trouvé
            Incrémentation de nbOcc dans la case nbOcc de ce mot
        Sinon création d'une nouvelle ligne
    Sinon on met 0 dans la case MotClef de ce mot
    On met 1 dans la case nbOcc de ce mot

Pour chaque mot-clef du texte source
Mise à jour du nombre d'occurrences du mot-clef par rapport au tableau d'occurrences
On cherche les occurrences du mot-clef dans le tableau d'occurrences
On met à jour l'occurrence du mot courant

- Sélection des phrases

Calcul du Score des phrases

Pour chaque mot (mot ou ponctuation) du texte source
    Si le mot courant est un mot
        Si le mot courant est un mot-clef
            On incrémente le score de la phrase qui le contient
        On incrémente le nombre de mots total

Sélection des phrases

Calcul de la contraction du résumé par l'équation :
Reste - (nombre de l'utilisateur * nombre total de mots) /100

Tant que Reste est supérieur ou égal à 0
    Pour chaque phrase du texte source
        Si la phrase est sélectionnée
            Si le score de la phrase courante est supérieur à celui déjà stocké
                On met dans le score max, le score courant
    On met T à la phrase au score le plus fort
    On stocke dans Reste, le nombre de mots total ôté de l'ex Reste

Astuces, Problèmes et Évolution du logiciel

Astuces

Ouverture du bon dictionnaire

Le dictionnaire se présentant sous la forme de : a.txt, b.txt, c.txt...etc, l'astuce a été ***d'ouvrir le dictionnaire grâce à la première lettre du mot recherché***. Par exemple, si le mot à recherché est "**a**beille", alors, la fonction RechercheDico ouvrira directement le dictionnaire **a**.txt

Filtre d'une lettre accentuée ou majuscule

Le logiciel ne pouvait pas traiter les mots commençant par une lettre accentuée (ex : **é**cole), car il ne trouvait pas le dictionnaire correspond à cette lettre accentuée. Ce problème se posait de la même façon pour les mots commençant par une majuscule.
La solution a été d'élaborer une ***fonction qui ne retournait que la lettre minuscule.***

Gestion des connecteurs logiques

D'après l'évaluation du logiciel suédois, il m'a semblé utile de faire une ***fonction qui gérait la recherche de connecteurs logiques dans le texte.*** Les connecteurs logiques sont des expressions ou des mots découpant un texte de façon logique (ex : Dans un premier temps, Ensuite, Par exemple, Ainsi, Pour conclure…).

Afin de mieux repérer d'éventuelles phrases introduites par ces fameux connecteurs logiques, j'ai décidé de créer un fichier nommé « connecteurs.txt ». D'autre part, il faut préciser que je n'ai choisi de gérer que les mots-connecteurs et non les expressions-connecteurs, car elles étaient plus difficilement identifiables (composées au minimum de 2 mots).

De même, je n'ai pris que ***les connecteurs logiques*** qui me semblaient ***les plus pertinents (conclusion, comparaison, approximation, illustration) :***

Conclusion	Comparaison	Approximation	Illustration
bref	comme	probablement	ainsi
donc	parallèlement	vraisemblablement	notamment
finalement			
ainsi			

Voici les mots-connecteurs répertoriés dans le fichier connecteurs.txt. En face de chacun d'eux, il y a leur valeur respective. Par exemple, il m'a semblé plus important de prendre en considération le connecteur « donc » (valeur de 100 points), censé introduire une phrase de conclusion résumant le texte, plutôt que le connecteur « ainsi » (valeur de 50 points), censé introduire une idée explicative.

bref 100
donc 100
finalement 100
ainsi 50
probablement 50
vraisemblablement 50
notamment 5
comme 5
parallèlement 5

Si la fonction a trouvé un connecteur logique existant dans ce fichier, alors **elle ajoute sa valeur au score de la phrase auquel il appartient**. De même que pour la fonction de recherche des mots-clés, cette fonction ne tient pas compte de la casse du mot, c'est-à-dire qu'elle fera pas de différence entre « Donc », « donc », et « DONC ».

Ajout d'un mot-clef

Si l'utilisateur désire ajouter **un mot qu'il estime comme pertinent par rapport au secteur d'activité auquel il appartient**, il suffit d'ouvrir le dictionnaire qui correspond à la 1ère lettre de ce mot. Par exemple, si le mot à rajouter est « **m**ail », il faut alors chercher à ouvrir le dictionnaire « **m**.txt » pour l'y recopier à la suite des autres (ou dans l'ordre alphabétique de tous les autres mots-clés).

Ajout d'un mot-connecteur

Si l'utilisateur désire ajouter **un mot-connecteur qu'il estime pertinent par rapport à la structuration d'informations des mails qu'il reçoit régulièrement**, il suffit d'ouvrir le fichier « connecteurs.txt », de l'y ajouter, et d'indiquer la valeur qu'on souhaite lui donner.

Cette opération pourrait être transposée de la même manière pour le cas où l'utilisateur souhaiterait ajouter dans le fichier connecteurs, non pas un mot-connecteur mais véritablement un mot qu'il estime être pertinent et capital dans ses mail et qu'il ferait suivre d'une valeur conséquente.

Calcul des occurrences

Choisissons un texte de base d'environ 5 lignes que l'on modifiera à notre gré 5 fois de suite, et tâchons de montrer le comportement du logiciel en ce qui concerne les occurrences :

1) Texte Source :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Bref, le corbeau est un animal généreux.

Finalement, il sait soigner, donner et garder l'amitié des gens
Le corbeau est un animal altruiste.
Fin.

1 bis) Texte Résumé :
Bref, le corbeau est un animal généreux.
Finalement, il sait soigner, donner et garder l'amitié des gens

Le logiciel a bien choisi les phrases introduites par des connecteurs logiques car il a considéré les valeurs des connecteurs logiques « bref » et « finalement » comme étant plus fortes que la valeur des mot-clefs seuls.

2) Texte Source :
Si mentir était l'apanage du **corbeau**, alors il faudrait se taire.
Le **corbeau** est un **animal** généreux.
Il sait soigner, donner et garder l'amitié des gens.
Finalement, le **corbeau** est un **animal** altruiste.
Fin.

2 bis) Texte Résumé :
Le **corbeau** est un **animal** généreux.
Finalement, le **corbeau** est un **animal** altruiste.

Le logiciel a pris la seule phrase introduite par un connecteur logique et une phrase comportant les mots « corbeau » et « animal » qu'il a détecté comme ayant le plus d'occurrences possibles dans le texte source.

3) Texte Source :
Si mentir était l'apanage du **corbeau**, alors il faudrait se taire.
Le **corbeau** est un animal généreux.
Il sait soigner, donner et garder l'amitié des gens.
Bref, le **corbeau** est un animal altruiste.
Fin.

3 bis) Texte Résumé :
Le **corbea**u est un animal généreux.
Bref, le **corbeau** est un animal altruiste.

Le logiciel a pris la seule phrase introduite par un connecteur logique et une phrase comportant les mots « corbeau » et « animal » qu'il a détecté comme ayant le plus d'occurrences possibles dans le texte source.

4) Texte Source :
Si mentir était l'apanage du **corbeau**, alors il faudrait se taire.
Le **corbeau** est un animal généreux.
Il sait soigner, donner et garder l'amitié des gens.
Donc, le **corbeau** est un animal altruiste.
Fin.

4 bis) Texte Résumé :

Le **corbeau** est un animal généreux.
Donc, le **corbeau** est un animal altruiste.

Le logiciel a pris la seule phrase introduite par un connecteur logique et une phrase comportant le mot « corbeau » qu'il a détecté comme ayant le plus d'occurrences possibles dans le texte. De plus, la réécriture du texte source dans le texte cible s'est passée correctement puisque l'ordre des phrases du texte cible correspond à l'ordre des phrases du texte source.

D'une manière générale, j'ai donc choisi **d'accorder plus d'importance à une phrase introduite par un connecteur logique** car cette dernière était statistiquement sûre de comporter des informations sur le sujet du texte. Ensuite, j'ai décidé **d'accorder de l'importance aux phrases contenant des mots-clefs ayant le plus d'occurrences** dans tout le texte. Enfin, j'ai donné une importance aux mots qui étaient contenus dans le dictionnaire de mots-clefs et qui se révélaient donc être des mots-clefs.

On pourrait schématiser la notion d'importance et de valeur attribuée aux mots et aux phrases par ces expressions :

Connecteur logique > Mots-clefs les plus récurrents > Mots-clefs > Mots

Phrase + Connecteur logique > Phrase + Mots-clefs les plus récurrents > Phrase + Mots-clefs > Phrase sans aucun mot-clef

Problèmes

Voici la liste des cas possibles qui ne pourront être traités de manière optimale par le logiciel :

- mails non accentués
- mails contenant des mots mal orthographiés
- mails contenant des mots composés
- mail ne contenant pas de points de fin de phrase
- mails contenant des phrases à caractère oral, avec des points de suspension (ex : je dis ça**…** parce que**…** cela fait si longtemps**…**)
- mails contenant des sigles (ex : E.N.S.T., C.N.R.S)
- mails contenant des mots d'origine étrangère (ex : mail, Windows, Microsoft)
- mails contenant des expressions-connecteurs (ex : En conclusion)
- mails contenant des mots de la même racine
- texte vide ayant juste un fichier attaché

Les mots non accentués ou mal orthographiés n'existent pas dans le dictionnaire. Donc, ***même si les mots seront traités, nous ne pourrons pas déterminer si ce sont des mots***

clés car ils ne seront pas comptabilisés au niveau de leurs occurrences éventuellement rencontrées dans le texte source.

A tout moment, il est possible pour l'utilisateur de rajouter directement dans le dictionnaire qui correspond, le mot qu'il désire voir affiché comme mot-clef.

Évolution

- **mails non accentués**

Solution : Il s'agirait ici d'élaborer un logiciel qui réaccentue les mails source, en faisant une recherche dans les dictionnaires.

- **mails contenant des mots mal orthographiés**

Solution : Nous pourrions envisager l'élaboration d'un correcteur d'orthographe qui corrige les fautes dans les mails source.

- **mails contenant des phrases à caractère oral, avec des points de ponctuation de fin de phrase**

Solution : Ici, il serait subtile de poser une démarcation logique entre l'oral et l'écrit dans des phrases « orales » écrites (ex : je me disais**…** « il a perdu ses clefs » **…**et c'était vrai**…**). Le but, difficile à atteindre, serait de pouvoir donner une approche cognitive à un module de découpage de phrases qui serait destiné à intervenir sur des phrases écrites compréhensibles à l'oral.

- **mails contenant des sigles**

Solution : Faire une fonction spéciale pour déterminer les sigles dont le point est différent de celui d'une fin de phrase. En effet, il faut que cette fonction vérifie par exemple, que le découpage d'une phrase ne va pas s'arrêter à :
« E**.** » au lieu de « E.N.S.T**.** »
ou
« E.D**.** » au lieu de « E.D.F »

- **mails contenant des mots d'origine étrangère (ex : mail, Windows, Microsoft)**

Solution : Soit trouver un dictionnaire de mots d'origine étrangère, soit ajouter à la liste des dictionnaires déjà existants, les dits mots-clefs.

- **mails contenant des mots composés**

Solution : Faire une fonction spéciale pour déterminer les mots composés

- **mails contenant des expressions-connecteurs**

Solution : Optimiser la recherche de mots-connecteurs déjà créée

- **mails contenant des mots de la même racine**

Solution : Il faudrait élaborer une fonction sachant repérer les mots ayant une racine susceptible de se retrouver dans d'autres mots. En effet, ces mots ont souvent la même définition.

Par exemple, le mot « enfant » se retrouve dans « enfance, enfanter, enfantement... » et veut dire quasiment à chaque fois la même chose puisqu'il s'agit de parler d'enfant. Donc, il faudrait avoir un dictionnaire réunissant ces racines qui comportent pour chacune d'elles, les déclinaisons de mots possibles à partir d'elles.

- **texte vide ayant juste un fichier attaché**

Solution : Créer une fonction qui va voir si le mail contient un corps de texte sinon, regardera si il existe une pièce jointe.

En conclusion, nous pouvons donc penser que le résumeur interviendrait dans la dernière phase du processus. C'est-à-dire que le logiciel de résumé ne pourrait résumer qu'à partir du moment où les logiciels (réaccentueur, correcteur d'orthographe...) auraient déjà scanné et modifié le texte source.

Pour ce qui est de la classification de messages, on pourrait par exemple s'inspirer de la recherche de mots clés déjà existante pour le résumeur. En effet, la classification de messages reviendrait à rechercher des messages contenant les mots clés jugés pertinents.

Connaissances acquises

Ce stage a été très formateur aussi bien sur le plan informatique que le plan linguistique.

Sur le plan Technique :

Je me suis formée à l'environnement UNIX sur des stations de travail Sun Solaris.

J'ai élaboré l'analyse de mon résumeur automatique en me basant sur des faits logiques et mathématiques tels que des statistiques de fréquences d'occurrences de mots clés. Puis, j'ai appris à utiliser à la fois l'éditeur Emacs sous UNIX et Visual C++ 6.0 sous Windows pour compiler mon code écrit en C.

Sur le plan Linguistique :

J'ai pu me mettre dans la peau d'un expert en évaluation de système de résumé automatique en analysant le logiciel de résumé automatique de textes développé par Hercules Dalianis et Martin Hassel. Ceci m'a permit de me confronter à la pertinence d'une évaluation d'un système donné et aux choix d'orientation à effectuer en constatant les résultats obtenus.

J'ai appris à me confronter aux problèmes d'intelligibilité et de cohérence que peuvent poser des résumés automatiques de textes.

Sur le plan humain :

J'ai découvert le monde de la recherche que je ne connaissais que partiellement, ayant toujours travaillé dans des entreprises. ***Grâce aux réunions sur l'avancée du projet MAJORDOME, j'ai réalisé l'efficacité du travail collectif et l'importance des qualités de relations entre les différents partenaires d'un projet de grande envergure.***

Ainsi, j'ai compris les enjeux que pouvaient comporter de gros projets de recherche *(délais, aspects financiers, communication entre les groupes de recherche internationaux et nationaux).*

Remerciements

Je remercie toute l'équipe du Département TSI qui m'a accueillie chaleureusement.

- Gérard CHOLLET pour m'avoir offert ce stage et été mon directeur de stage.
- Pascal VAILLANT pour sa bonne humeur et pour m'avoir conseillée.
- François YVON pour m'avoir guidée dans l'orientation de mes lectures.
- Laurence LIKFORMAN pour m'avoir suivie dans la progression de mon logiciel

Bibliographie et Annexes Techniques

Bibliographie

- **« Evaluating Natural Language Processing Systems »**
  K.Sparck Jones/Julia R.Galliers

- **« L'activité résumante »**
  M.Charolles & A.Petitjean *(Collection Didactique des Textes)*

- **« Perl Reference Manual version 5.002 beta 1 g »**
  Bilbo Baggins, in Perl.c

Webographie

www.nada.kth.se/~xmartin/swesum/index-eng.htm

www.itl.nist.gov/iaui/89402/related-projects/tipster/gen-ie.htm

http://www.lehmam.freesurf.fr/autoresu.htm

http://www.cnam.fr/instituts/INTD/forum/site/resume/resume2.html

http://m17.limsi.fr/RS96FF/CHM/LC/LC8.html

http://www.slis.ualberta.ca/cais2000/balicco.htm

http://palf.free.fr/esaintot/connecteurs.htm

Annexes Techniques

Annexe A

- **Rapport d'activité : Evaluation du Système de Résumé automatique de textes "SWESUM" de Messieurs Hercules DALIANIS & Martin HASSEL**
  Aude Acoulon

Annexe B

- **Mode d'emploi pour le logiciel de résumé automatique**
  Aude Acoulon

Annexe C

- **Prototype des fonctions utilisées pour le logiciel de résumé automatique**
  Aude Acoulon

Annexe D

- **Comparaison des logiciels de résumé automatique de textes "Swesum" & "Resume"**
  Aude Acoulon

Annexe E

- **« Description of human language technology products »**
  Hercules Dalianis NADA-KTH

- **« Automatic writing : Text Generation and Summarization »**
  Hercules Dalianis NADA-KTH

- **« Advanced in Automatic text summarization »**
  K.Sparck Jones

- **« Swesum A text Summarizer for Swedish »**
  Hercules Dalianis NADA-KTH

- **« Development of a Swedish Corpus for Evaluating Summarizers and other IR-tools »**
  Hercules Dalianis and Martin Hassel NADA-KTH

- **« Improving Precision in Information Retrieval for Swedish using Stemming »**
  Johan Carlberger, Hercules Dalianis, Martin Hassel, Ola Knutsson NADA-KTH

- **« Ingénierie des Langues »**
  sous la direction de Jean-Marie Pierrel *(Hermes Science publications)*

- **« The TIPSTER SUMMAC Text Summarization Evaluation-Final Report »**
  The MITRE Corporation

ANNEXE A

EVALUATION
DU
LOGICIEL DE RESUME AUTOMATIQUE DE TEXTES
« SWESUM »
de
Messieurs Hercules DALIANIS & Martin HASSEL,
partenaires Suédois du projet Majordome

Rapport d'activité du 20 septembre 2001
par Aude ACOULON
sous la direction de G.CHOLLET, P.VAILLANT

SOMMAIRE

PRESENTATION DE L'EVALUATION 4
Les Catégories 5
Critères d'Évaluation 6
Type de Fiche de travail 6
Explications de la Fiche 6
Conditions de travail 7
EVALUATION DES CATEGORIES 8
Les Dépêches d'agence 8
Les Forums de Discussion 8
Les Mails 10
Les Fax 11
CONCLUSION 12

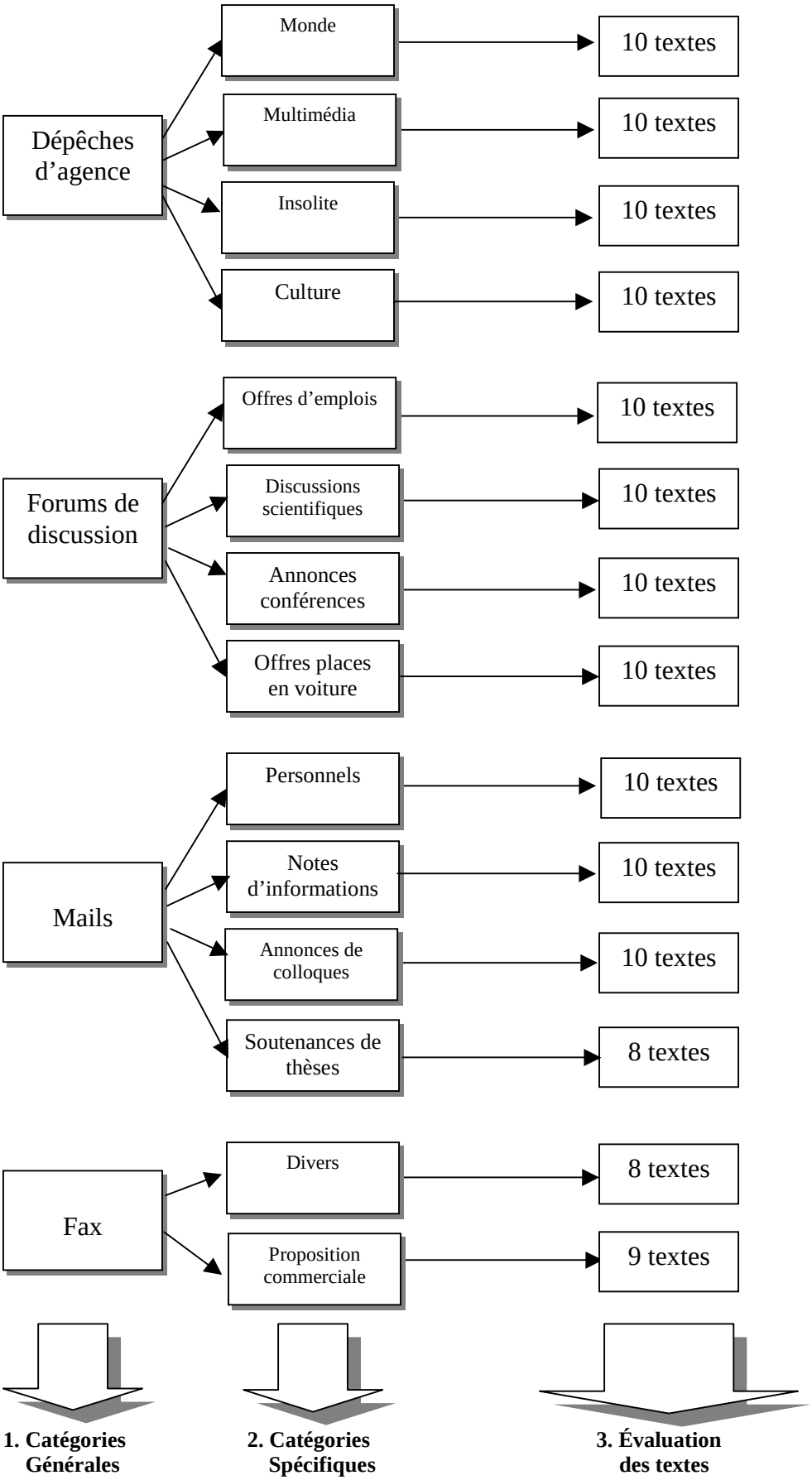
PRESENTATION DE L'EVALUATION

Il a été établi quatre sortes de textes :

- dépêches d'agence
- forums de discussion
- mails
- fax

Dans chaque catégorie, 3 ou 4 sous-catégories ont été créées. Puis, pour chaque texte, une petite fiche fait la comparaison entre le texte source et le texte cible à travers les différents critères d'évaluation choisis.

Les Catégories



NB : il a été difficile de retenir des fax susceptibles d'être résumés par le logiciel suédois. Néanmoins, je me suis efforcée de classer en 2 sous-parties, les fax les plus lisibles.

Au total, j'ai pu procéder à ***135 fiches d'évaluation.***

Critères d'Évaluation

L'évaluation de la démo s'est effectuée selon divers critères au niveau des résumés obtenus. Je me suis donc servi d'une **fiche de travail**, type fiche signalétique pour rendre compte de l'essentiel de chaque texte évalué.

Type de Fiche de travail

Fiche numero :

Nombre de mots du texte source :
Nombre de mots du texte cible :
Contraction du resume : %
Type de texte : journal ou académique
Mot(s) cle(s) :

Informations

Qui : oui/moyen/non/pas indique dans le texte source
Quoi : oui/moyen/non/pas indique dans le texte source
Ou : oui/moyen/non/pas indique dans le texte source
Comment : oui/moyen/non/pas indique dans le texte source
Pourquoi : oui/moyen/non/pas indique dans le texte source

Contexte : OK (texte source = texte cible), partiellement OK, informations oubliées
Longueur : bonne/moyenne/trop court…
Intelligibilite : bonne/moyenne/peu claire

Remarque(s) :

Explications de la Fiche

Informations :
Parfois, les réponses aux questions primaires n'étaient pas présentes dans le texte source. Dans ce cas, je l'indiquais entre parenthèses ou directement.

Contexte :
Si le contexte du résumé reflétait point à point l'idée maîtresse du texte source alors je l'indiquais en notant : OK.
Si le résumé comportait les idées générales du texte source sans les expliquer alors je notais : partiellement OK.
Si le logiciel n'avait pas su redonner toutes les idées dans le résumé alors je notais : informations oubliées.

Longueur :
Je prenais comme critère le taux de contraction du résumé produit par rapport au texte source.
Si la contraction était autour des 30%, je notais : bonne.
Si la contraction était moins de 20%, je notais : un peu trop court.
Si la contraction était autour de 40%, je notais : moyenne.
Si la contraction était de plus de 50%, je notais : un peu trop long.

Intelligibilité :
J'observais la cohérence du résumé produit, en remarquant par exemple si des phrases contradictoires étaient collées côte à côte ou si lors de la lecture, la logique de pensée du résumé était fluide.

Remarque(s) :
Je notais ici, ce que je ne pouvais écrire ailleurs : les interrogations ou réflexions que j'avais pu tirer de la production d'un résumé.

Conditions de travail

Les conditions dans lesquelles je travaillais étaient les suivantes :

- Taux de contraction pour le résumé : 30%
- Textes français à résumer en français
- Résumé de style journalistique plutôt qu'académique
- Affichage des mots-clefs reconnus dans le texte source

Lien du Logiciel : http://www.nada.kth.se/~xmartin/swesum/index-eng-adv.html

EVALUATION DES CATEGORIES

Les Dépêches d'agence

Dans la majorité des cas, **le logiciel a su redonner de façon partielle un contexte** à chacun des résumés produits.

Lorsqu'il existe dans le texte source, **une liste d'informations relatives à des données d'ordre économique (pourcentages, taux,...) ou encore une liste de noms propres,** le logiciel, qui ne recherche que les mots-clefs, va retenir des phrases-clefs (c'est-à-dire contenant le plus de mots-clefs) mais aussi les données d'ordre économique. Par conséquent, le résumé produit nous présente des informations économiques disparates, parfois contradictoires parce que simplement contenues dans des phrases jugées pertinentes par le logiciel, mises côte à côte.

Des phrases longues dans le texte source entraînent une mauvaise cohérence dans le texte cible. En effet, le logiciel ne prend alors en compte que des phrases jugées, statistiquement être les plus pertinentes, sans soucier de cohérence entre les phrases.

Quand un texte source ne contient pratiquement que des exemples, le logiciel nous présente un résumé ayant perdu beaucoup d'informations souvent introduites par des connecteurs logiques servant à illustrer une idée (ex : ainsi, par exemple, d'ailleurs, notamment). **Comme ces connecteurs n'ont pas de « valeur » reconnue et qu'ils sont juste répertoriés comme simples mots-clefs, la cohérence entre les phrases choisies pour le résumé est moyenne.**

D'autre part, **lorsqu'un texte source ne contient que trois longues phrases, le logiciel résume alors ces phrases avec une contraction de 75%, car le texte n'offre pas un assez grand nombre de phrases.** En sortie, le texte résumé est quasiment la copie conforme du texte source, car le nombre de phrases est, d'une manière générale, proportionnel au nombre d'idées différentes d'un texte. Par conséquent, plus un texte a de phrases, plus un choix entre les phrases à ne retenir pour le résumé, peut être fait. En revanche, la probabilité d'obtenir un résumé « cohérent » ou « compréhensible » est plus faible, compte tenu du choix des phrases mises côte à côte. Peut-être faudrait-il imposer une limite maximum de contraction de résumé afin d'éviter une copie conforme entre le texte source et le texte cible ?

Quand un texte source contient une image et une explication en-dessous de cette image, le logiciel prend cette explication comme faisant partie intégrante du texte source et, l'intègre (si elle présente un mot-clef) dans le résumé final, ce qui crée des problèmes de cohérence entre les phrases du résumé.

Les Forums de Discussion

Pour le cas d'un texte sorti d'un forum de discussion, **le meilleur résumé obtenu est celui qui reprend la réponse d'un internaute faisant référence à une question posée ultérieurement dans le forum.**

Le plus mauvais résumé obtenu est celui d'un cas où un internaute a volontairement répondu sur un des éléments d'une question posée par un internaute. Dans le texte source, l'internaute n'avait répondu que partiellement à la question posée sur le forum (qui elle, n'est pas incluse dans le résumé). Par conséquent, **les phrases choisies pour le résumé et mises côte à côte, manquent de cohérence et il est difficile de pouvoir en redonner un contexte bien précis.**

Dans la majorité des cas, le logiciel ne prend pas en compte **les informations essentielles** car celles-ci sont **séparées, le plus souvent, par des séries d'espaces** (ou tabulations) normées indiquant des informations courtes telles qu'un titre, une date... Le résumé obtenu ne nous donne alors, aucune information sur l'objet même de la discussion car **la « reconnaissance d'une phrase » n'est faite que lorsque le logiciel rencontre une ponctuation de fin de phrase (? . ; ! ...).** Ce découpage peut se révéler être gênant quand il s'agit d'une annonce de thèse par exemple, où il manque les champs et leurs données essentielles :

Titre :
Candidat :
Lieu :
Date :

Parfois, le logiciel indique de fausses informations, comme une contraction de seulement 18% pour un texte source composé de 124 mots. Le résumé obtenu apparaît alors superficiel car absolument pas pertinent du point de vue des statistiques. En effet, pour un texte source contenant 124 mots, le logiciel doit être à même de pouvoir réaliser une contraction de 30%, contraction demandée par l'utilisateur du logiciel, au départ.

Par ailleurs, **des fautes d'orthographe ou de grammaire dans le texte source** (ex : *"l'utilité des cellules embryonnaires **et** qu'on peut les faire **devnir** ce qu'on veut"*) **nuisent au bon repérage des mots-clefs et quelque peu au contexte.**

Quand il s'agit d'un **forum de discussion scientifique, la cohérence d'un résumé devient capitale.** Dans certains textes, le logiciel "tronque" des segments de phrases-clef. La lecture du résumé n'a plus de fil conducteur, de logique dans l'argumentation et **on ne peut plus retrouver la manière de penser de l'auteur.**

Parfois, au contraire, le résumé est quasiment une copie conforme du texte source. Par conséquent, la logique de pensée et la cohérence entre les phrases est évidemment excellente.

Les acronymes ou les sigles ne sont pas « reconnus » par le logiciel (surtout s'ils n'apparaissent qu'une seule fois), **même si l'objet du mail en dépend** (ex : une annonce d'emploi décrivant *un poste en **CDI** ou en **CDD** demandant lettre et **CV***). De même, **le logiciel ne prend pas en compte les mots nouveaux ou étrangers** (ex : *Newsgroups, From, To, Subject*).

Le logiciel ne sait pas gérer de trop courtes et peu nombreuses phrases. Dans la plupart des cas, la contraction du résumé révèle 100%, ce qui est faux étant donné le nombre de mots du texte source comparé au nombre de mots du texte cible... Parfois, **le logiciel ne**

trouve aucun mot-clef, ce qui semble être dû à la trop courte longueur du texte source, ne donnant pas assez de statistiques pour repérer les éventuels mots-clés.

Les Mails

D'une manière générale, le logiciel n'offre pas un bon résumé pour les mails. En effet, il y a **beaucoup d'informations inutiles** qui ressortent du résumé alors que les informations pertinentes manquent cruellement à l'identification du contexte.

La plupart du temps, **le logiciel ne supporte pas une abondance de petits signes stylistiques que l'on peut trouver dans les mails** (tirets rapprochés spécifiant un trait de soulignement, étoiles, ...). Afin de faire résumer le texte source au logiciel, il faut tout d'abord, procéder à la suppression de ces signes pour obtenir un résumé.

Lorsqu'il y a absence de ponctuation de fin de phrase, le logiciel a tendance à prendre en compte **tous les mots quels qu'ils soient comme appartenant à une unique phrase**, ce qui pose des **problèmes de cohérence entre les phrases**. Par exemple, le logiciel peut reprendre dans un résumé, toute une liste de coordonnées téléphoniques de personnes ou encore de noms de participants à un colloque dont il n'est vraisemblablement pas utile de mentionner. La contraction du résumé se révèle alors souvent trop longue et la compréhension du texte cible, moyenne.

Toutefois, **le logiciel sait garder pour un résumé, les chevrons ou séries de chevrons symbolisant la redite ou d'éventuelles réponses aux réponses d'anciens mails**, ce qui améliore la lecture du mail.

Parfois il manque **des informations essentielles comme le lieu ou la date car celles-ci n'apparaissent qu'une fois et très fréquemment hors phrases**, ce qui peut s'avérer gênant en ce qui concerne des mails de colloques ou d'annonces de réunions. Ainsi, dans un mail annonçant une thèse, on peut remarquer que **le résumé obtenu n'est pas pertinent puisqu'il ne contient pas de réponses aux questions primaires** (qui, quoi, où, quand, comment). Par exemple, le résumé peut nous présenter des phrases issues du milieu du résumé de la thèse ! C'est pourquoi dans les mails d'annonces de thèse ou autre, nous n'avons généralement aucune information sur l'objet de la thèse elle-même, ni sur son auteur, ni sur son lieu...

D'autre part, le logiciel ne sait pas toujours relever pour le résumé, **des informations commençant par des connecteurs logiques** (par exemple, il ne connaît pas l'expression "*dans un deuxième temps*" alors qu'il prend en compte "*dans un premier temps*" car c'est une expression contenue dans une phrase-clef. De ce fait, il manque **une partie d'informations introduites par ces fameux connecteurs logiques** qui sont **nécessaires à la bonne intelligibilité du résumé.**

Pour ce qui est des mails personnels, le système le plus judicieux à adopter serait de prendre en considération pour destinataire, le/les nom(s) propre(s) après la formule de politesse bien connue "*Cher, Chère...*"et pour expéditeur, le/les nom(s) propre(s) après la formule "*Bien amicalement, Cordialement...*".

La présentation du mail elle-même peut faire défaut au résumé. Par exemple, le logiciel ne peut indiquer le nom de l'auteur d'une thèse si celui-ci est à la fin du mail, seul, en gage de signature.

Les Fax

D'une manière générale**, le logiciel ne sait pas résumer des textes sources peu lisibles**. De ce fait, **le résumé produit ne redonne pas un contexte aussi précis qu'il devrait et demeure très incohérent au niveau syntaxique.** De même, le logiciel ne comptabilisant seulement les mots qu'il « reconnaît », **le nombre de mots du texte source et le nombre de mots-clés sont souvent très bas.** Voici un exemple de résumé produit :

r& ~ est l'a vei';ture de C/ara, /`hérôûw, celle de tous les coeurs battants on c' ansons, musiqué, danse.. Tkibd tvS b `Il G: L'Y ~1 5E :ÇT SàS-~ - LT

D'autre part, **le logiciel ne prend pas en compte les champs essentiels d'un fax et les données qui s'y rapportent** (date, heure, page, expéditeur, destinataire, objet). Ainsi, si un fax nous explique la nature même de l'objet à l'intérieur du corps du message et que ce dernier est illisible, alors le résumé obtenu est incohérent et incompréhensible puisque nous ne pouvons connaître pas la nature du contexte. Peut-être, le logiciel pourrait uniquement considérer ce qu'il y a écrit après l'intitulé "*objet :* "

Cependant mais cela reste rare, le logiciel produit **un assez bon résumé, reprenant les phrases-clef qui, pourtant, comportaient des éléments illisibles au départ**. **Le résultat en terme de cohérence et d'intelligibilité n'est pas parfait mais apparaît honorable** si l'on tient compte de l'illisibilité de départ.

Les dates sous forme de jj:mm:aa ne sont pas interprétables pour le logiciel qui ne les reconnaît pas comme étant des mots ou des mots-clés.

Dans la plupart des cas, **le taux de contraction est faux car le nombre de mots du texte source est faux** (le logiciel ne comptant que les mots qu'il reconnaît). Par exemple, si un résumé obtenu est incohérent ou incompréhensible et, si le logiciel indique une contraction de 100% c'est parce qu'il aura su « recopié » les seuls mots qu'il avait « reconnus ».

Donc, dans tous les cas de figure, **par manque de lisibilité**, on peut penser que **le comptage des mots est faux**. Si un texte source ne comporte pas assez de mots lisibles pour produire un résumé alors le logiciel « reconstruit un résumé » avec les bribes de mots qui lui restent. Par conséquent, **il n'est pas rare de rencontrer des résumés composés de mots seuls, souvent trop peu nombreux pour pouvoir rétablir un contexte initial**, comme l'illustre nettement cet exemple de résumé de fax :

notre adresse e-mail, Monsieur DAUPHIN Gilles NUMERO FAX : UI 4608793 NOMBRE DE PAGES : 5

CONCLUSION

(déjà énoncée dans le rapport d'activité sur le Projet MAJORDOME) :

Les résultats ont divergé aussi bien selon la longueur des textes que leur contenu.

En effet, lorsqu' un texte court est composé d'une grande et d'une petite phrase, alors le logiciel ne prend en compte que la plus grande, celle qui contient le plus de mots-clés et non la plus petite même si celle-ci résumait le texte en son entier. Les connecteurs logiques découpent un texte en plusieurs parties et introduisent souvent les premières idées d'un paragraphe. Ainsi, la recherche de connecteurs logiques introduisant certaines phrases (ex : Donc, Bref, Finalement...) dites phrases-clés pourrait se révéler judicieuse. Cela pourrait permettrait par exemple, de **donner une plus grande valeur aux phrases, même courtes**, introduite par ces connecteurs, d'être sélectionnées pour la production d'un résumé.

D'une manière générale, **le logiciel ne peut reconnaître deux phrases différentes si une ponctuation de fin de phrase a été omise entre les deux**. Ce cas pourrait aussi se présenter dans la présentation d'informations séparées par une série d'espaces ou de tabulations. Le résumé obtenu nous présente alors une grande phrase contenant plusieurs mots-clés mais hélas incohérente puisque c'est la seule.

D'autre part, **le logiciel ne supporte pas les ponctuations stylistiques que l'on peut rencontrer dans des mails** (par exemple, des tirets mis côte à côte comme pour un trait de soulignement dans les annonces de séminaires, de thèses...). Similairement, les fax passés sous la technique de l'OCR contiennent des phrases peu lisibles que le logiciel ne prend pas en compte puisqu'il ne sait pas les corriger. Le résumé obtenu est difficilement compréhensible, tant au niveau syntaxique que sémantique.

En fait, le logiciel n'est pas capable de compenser ou de faire le travail d'un autre logiciel, à savoir, "reconnaître" des suites de mots appartenant (sémantiquement, syntaxiquement) ou non à une même phrase afin de passer à la phrase suivante.

En conclusion, nous pouvons donc penser que le logiciel ne pourrait s'exécuter qu'à partir du moment où d'autres logiciels existants (réaccentueur, correcteur d'orthographe...) auraient déjà scanné et modifié le texte source. Ainsi, le logiciel n'interviendrait que dans la dernière phase du processus.

ANNEXE B

MODE D'EMPLOI
Du Logiciel de Résumé Automatique

de
Aude ACOULON

---

27 septembre 2001
par Aude ACOULON
sous la direction de G.CHOLLET, P.VAILLANT

Utilisation du Logiciel

Pour utiliser le logiciel, il faut se placer dans un répertoire contenant :

- l'executable resume
- les dictionnaires en « a.txt », « b.txt », « c.txt »...
- le texte source en fichier.txt

Ensuite, derrière le prompt, il suffit de taper « **resume** » pour se rappeler la **syntaxe de la commande** :

```
<geebee:acoulon 7> resume
SYNTAXE : resume NomFichierSource NomFicherDest NomFichierConnecteur
[TauxContraction(1-99%) (defaut:30%)]
```

Il faut donc entrer à la suite, et **suivi d'un espace** :

- resume
- source.txt
- resume.txt
- connecteurs.txt
- 40 ou rien

comme le montre cet encadré :

```
<geebee:acoulon 8> resume texte41.txt texte41R.txt connecteurs.txt 40
```

ⓘ Le **taux de contraction** par défaut d'un résumé est de 30%. Il est indiqué entre crochets dans la syntaxe car il est **optionnel** : si l'utilisateur l'omet et qu'il appuie sur la touche espace, alors le logiciel interprète la contraction donnée par défaut (30%) :

```
<geebee:acoulon 8> resume texte41.txt texte41R.txt connecteurs.txt
NomFichierSource : (texte41.txt)
NomFichierDest : (texte41R.txt)
NomFichierConnecteurs : (connecteurs.txt)
TauxContraction : (30)
```

Après avoir rappelé à l'utilisateur les données entrées, le logiciel va **vérifier l'existence du nom du fichier entré**. S'il ne le trouve pas, il donne un message d'erreur du type « **Fichier inexistant** » « **Erreur module 1** ». Il faut alors recommencer le processus depuis le début.

En revanche, si le logiciel trouve le fichier, alors il va s'exécuter automatiquement en donnant le détail de ce qu'il est en train de calculer :

```
Module 1 running ...

Fichier trouve
Module 1 finished...
reduction du texte de 70% en cours ...
Module 2 running ...
Module 2 finished ...
Module 3 running ...
ScorePhrases  running ...
ScorePhrases  finished ...
SelectionPhrases  running ...
La phrase (0) avec un Score de:(7.583333) n'est pas selectionnee
La phrase (1) avec un Score de:(27.285715) est selectionnee
La phrase (2) avec un Score de:(17.363636) est selectionnee
La phrase (3) avec un Score de:(13.000000) n'est pas selectionnee
SelectionPhrases  finished ...
Reecriture  running ...
Reecriture  finished ...
Module 3 finished ...
<geebee:acoulon 9>
```

Ici, le logiciel a donc dû découper 4 phrases (noté 0,1,2,3) auquel il a attribué un score (noté entre parenthèses). La phrase est alors sélectionnée ou pas selon son score.

A présent, il suffit d'ouvrir le fichier créé dans le répertoire courant pour en apprécier les résultats.

Astuces et Fonctionnement du Logiciel

Ouverture du bon dictionnaire

Le dictionnaire se présentant sous la forme de : a.txt, b.txt, c.txt...etc, la première astuce est de pouvoir ***ouvrir le bon dictionnaire grâce à la première lettre du mot recherché***. Par exemple, si le mot à recherché est "**a**beille", alors, une fonction du programme ouvrira directement le dictionnaire **a**.txt

Filtre d'une lettre accentuée ou majuscule

Au départ, le logiciel ne pouvait pas traiter les mots commençant par une lettre accentuée (ex : **é**cole), car il ne trouvait pas de dictionnaire existant, correspondant à cette lettre accentuée. Ce problème se posait de la même façon pour les mots commençant par une majuscule.

La solution a été d'élaborer une fonction qui ne retournait que la lettre minuscule. Ainsi, **si vous désirez résumer un mail accentué ou écrit entièrement en lettres capitales**, vous pouvez le faire sans aucune difficulté. En revanche, il n'est guère possible pour l'instant, de pouvoir résumer des mails NON accentués à la base.

Gestion des connecteurs logiques

Les connecteurs logiques sont des expressions ou des mots découpant un texte de façon logique (ex : Dans un premier temps, Ensuite, Par exemple, Ainsi, Pour conclure…).

Afin de mieux repérer d'éventuelles phrases introduites par ces fameux connecteurs logiques, il existe un fichier nommé « connecteurs.txt ». Il faut préciser que **le logiciel ne sait gérer que les mots-connecteurs** et non les expressions-connecteurs, car ces dernières sont plus difficilement identifiables (composées au minimum de 2 mots).

Par défaut, dans le fichier « connecteurs.txt », il existe ***les connecteurs logiques*** apparaissant ***les plus pertinents (conclusion, comparaison, approximation, illustration) :***

Conclusion	Comparaison	Approximation	Illustration
bref	comme	probablement	ainsi
donc	parallèlement	vraisemblablement	notamment
finalement			
ainsi			

En face de chacun de ces mots-connecteurs, vous trouverez dans le fichier « connecteurs.txt », leur valeur respective. Par exemple, le connecteur jugé le plus important

est le connecteur « donc » a une valeur de 100 points car il est censé introduire une phrase de conclusion résumant le texte.Le connecteur « ainsi » jugé moins important a une valeur de 50 points, car il est censé introduire une idée explicative.

bref 100
donc 100
finalement 100
ainsi 50
probablement 50
vraisemblablement 50
notamment 5
comme 5
parallèlement 5

Si la fonction a trouvé un connecteur logique existant dans ce fichier, alors **elle ajoute sa valeur au score de la phrase auquel il appartient**. De même que pour la fonction de recherche des mots-clés, cette fonction ne tient pas compte de la casse du mot, c'est-à-dire qu'elle fera pas de différence entre « Donc », « donc », et « DONC ».

Ajout d'un mot-clef

Si vous désirez ajouter **un mot que vous estimez comme pertinent par rapport au secteur d'activité auquel vous appartenez**, il suffit d'ouvrir le dictionnaire qui correspond à la 1ère lettre de ce mot. Par exemple, si le mot à rajouter est « **m**ail », il faut alors chercher à ouvrir le dictionnaire « **m**.txt » pour l'y recopier à la suite des autres (ou dans l'ordre alphabétique de tous les autres mots-clés).

Ajout d'un mot-connecteur

Si vous désirez ajouter **un mot-connecteur que vous estimez pertinent par rapport à la structuration d'informations des mails que vous recevez régulièrement**, il suffit d'ouvrir le fichier « connecteurs.txt », de l'y ajouter, et d'indiquer la valeur que vous souhaitez lui donner.

Cette opération pourrait être transposée de la même manière pour le cas où vous souhaiteriez ajouter dans le fichier connecteurs, non pas un mot-connecteur mais véritablement un mot que vous estimez pertinent et capital dans vos mail et que vous feriez suivre d'une valeur conséquente.

ⓘ A tout moment, il est possible pour l'utilisateur de rajouter directement dans le dictionnaire qui correspond, le mot qu'il désire voir affiché comme mot-clef.

ANNEXE C

```
/* resume.h */
#include <stdio.h>
#include <stdlib.h>
#include <string.h> /* pour strcpy */
#include <ctype.h> /* pour isalpha */


    /* cette structure contiendra les mots et leurs attributs */

    typedef struct Element
     {
     unsigned char Mot[255];            /* contient le mot ou une ponctuation */
                                        /* 255 pour les tests voir avec malloc apres */
     struct Element *ElementSuivant;    /* contient un pointeur vers le mot suivant */
     struct Element *ElementPrecedent;  /* contient un pointeur vers le mot precedent */
     int nbOcc;                         /* contient le nombre d'occurrences du mot */
     int noPhrase;                      /* contient le numero de la phrase a laquelle
                                        appartient le mot */
     int noParagraphe;                  /* contient le numero de paragraphe auquel
                                        appartient le mot */
     char MotClef;                      /* contient un mot-clef T(true) sinon F(false) */
    }tabElement;


    /* cette structure contiendra les occurrences des mots cles */

    typedef struct TOCC
     {
     unsigned char Mot[26];             /* contient le mot-clef et ses attributs */
     int nbOcc;                         /* contient le nombre d'occurrences du mot-clef */
     }OCC;

     typedef struct TABOCC
    {
          OCC * tabOcc;                 /* tableau d'occurrences */
          int nbMotsCles;               /* nombre de mots-cles trouves */
     }TAB_OCC;



    /* cette structure contiendra les scores des phrases */

    typedef struct TSCORE
    {
     int noPhrase;                      /* contient le numero de la phrase */
     float ScoreMotsCles;               /* contient le score des mots-cles de la phrase */
     int nbMots;                        /* contient le nombre de mots de la phrase */
     char Flag;                         /* contient un booleen T(true) ou F(false)indiquant
                                        si la phrase est a selectionner */
            /*int ScoreMax;            /* contient T si la phrase est a reecrire */
     }SCORE;


    /* Definition des prototypes des fonctions utilisees */
```

```
/*****************************************************************************/
/* VerifFichier                                                              */
/* -------------                                                             */
/* Entree :                                                                  */
/*  - Nom du fichier dont il faut verifier les droits en  ecriture. Type : char*  */
/* Sortie :                                                                  */
/*  - 1 si le fichier peut etre ecrit, 0 sinon.                              */
/* Autres fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va verifier s'il est posssible de créer le fichier avec le nom donne  */
/*****************************************************************************/
int VerifFichier(char*);


/*****************************************************************************/
/* affichage                                                                 */
/* ---------                                                                 */
/* Entree :                                                                  */
/*  - Pointeur sur la structure ou va etre stocke le texte.                  */
/*    Type Element*                                                          */
/* Sortie :                                                                  */
/*      Aucune                                                               */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va afficher tous les mots du texte. Les mots-cles seront precedes d'une etoile.  */
/*****************************************************************************/
void affiche(tabElement **);



/*****************************************************************************/
/* module 1                                                                  */
/* --------                                                                  */
/* Entree :                                                                  */
/*  - Pointeur sur la structure ou va etre stocke le texte.                  */
/*    Type Element*                                                          */
/*  - Nom du fichier contenant le texte source. Type : char*                 */
/* Sortie :                                                                  */
/*  - 0 si tout s'est bien passe, 1 sinon. Le seul cas  pouvant poser probleme est l'inexistence ou  */
/*    la non-accessiblite du fichier source                                  */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va lire le fichier source puis va stocker les differents mots, ponctuations,  */
/* espaces, grace a la  structure passee en argument                         */
/*****************************************************************************/
int module1(tabElement **, char *);
```

```
/*****************************************************************************/
/* module 2                                                                  */
/* --------                                                                  */
/* Entree :                                                                  */
/*  - Pointeur sur la premiere structure composant le texte.                 */
/*    Type : tabElement*                                                     */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  - RechercheDico                                                          */
/*  - cptOccMotClef                                                          */
/* Cette fonction va rechercher dans le dictionnaire si le mot est un mot-clef et quelles sont ses  */
/* occurrences connues dans le texte source                                  */
/*****************************************************************************/
void module2(tabElement **);


/*****************************************************************************/
/* RechercheDico                                                             */
/* --------------                                                            */
/* Entree :                                                                  */
/*  - Pointeur sur le mot dont il faut rechercher l'existence dans le dictionnaire.  */
/*     Type : tabElement*                                                    */
/*  - Pointeur sur le tableau d'occurrences de mots-cles. Type : TAB_OCC     */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  - FiltreLettre                                                           */
/* Cette fonction va rechercher dans le dictionnaire si le mot y est contenu : si c'est le cas,      */
/* alors le mot est un mot-clef, la fonction retourne 0 ou 1                 */
/*****************************************************************************/
  int RechercheDico(tabElement **,TAB_OCC *);


/*****************************************************************************/
/* FiltreLettre                                                              */
/* -------------                                                             */
/* Entree :                                                                  */
/* - lettre accentuee ou lettre majuscule                                    */
/* Sortie :                                                                  */
/* - lettre non accentuee ou lettre minuscule                                */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction ne tient pas compte de la casse ni des accents et renvoie la lettre minuscule   */
/* ou non accentuee                                                          */
/*****************************************************************************/
char FiltreLettre(char);
```

```
/*****************************************************************************/
/* cptOccMotClef                                                             */
/* --------------                                                            */
/* Entree :                                                                  */
/*  - Pointeur sur le texte dont on doit mettre a jour le nombre d'occurrences.  */
/*    Type : tabElement*                                                     */
/*  - Pointeur sur le tableau d'occurrences de mots-cles. Type : TAB_OCC     */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va calculer le nombre d'occurrences du mot-clef et mettre a jour les */
/* occurrences de chaque mot-clef du texte                                   */
/*****************************************************************************/
void cptOccMotClef(tabElement *, TAB_OCC *);


/*****************************************************************************/
/* module 3                                                                  */
/* --------                                                                  */
/* Entree :                                                                  */
/*  - Pointeur sur la premiere structure composant le texte. Type : tabElement*  */
/*  - nomFichierDest Type : char*                                            */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  - ScorePhrases                                                           */
/*  - SelectionPhrases                                                       */
/*  - Reecriture                                                             */
/* Cette fonction va selectionner les phrases les plus fortes pour etre reecrites dans le texte */
/* cible                                                                     */
/*****************************************************************************/
void module3(tabElement **, char*);


/*****************************************************************************/
/* ScorePhrases                                                              */
/* -------------                                                             */
/* Entree :                                                                  */
/*  - Le texte source Type : tabElement*                                     */
/*  - t_Score Type : SCORE*                                                  */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va calculer le score de chaque phrase en comptant les occurrences des */
/* mots-cles qu'elle contient                                                */
/*****************************************************************************/
void ScorePhrases(tabElement **, SCORE *);
```

```
/*****************************************************************************/
/* SelectionPhrases                                                          */
/* ----------------                                                          */
/* Entree :                                                                  */
/*  - t_Score Type : SCORE*                                                  */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va selectionner les phrases aux scores les plus forts et remettre a jour le */
/* tableau de scores des phrases                                             */
/*****************************************************************************/
void SelectionPhrases(SCORE *);


/*****************************************************************************/
/* Reecriture                                                                */
/* ----------                                                                */
/* Entree :                                                                  */
/*  - Phrase ayant un score fort.                                            */
/*    Type : tabElement*                                                     */
/*  - Le tableau de score de phrases. Type : SCORE *                         */
/*  - Le fichier destination                                                 */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va recopier les phrases aux scores selectionnees en conservant l'aspect */
/* initial du texte source                                                   */
/*****************************************************************************/
void Reecriture(tabElement**,SCORE *,FILE *);


/*****************************************************************************/
/* RechercheConnecteur                                                       */
/* -------------------                                                       */
/* Entree :                                                                  */
/*  - Le tableau de score de phrases. Type : SCORE *                         */
/* Sortie :                                                                  */
/*  Aucune                                                                   */
/* Autres Fonctions utilisees :                                              */
/*  Aucune                                                                   */
/* Cette fonction va rechercher s'il existe un connecteur dans le texte source. Puis elle va */
/* mettre a jour le score de la phrase avec la valeur du connecteur logique  */
/*****************************************************************************/
 int RechercheConnecteur(const char * mot);
```

ANNEXE D

COMPARAISON
DES
LOGICIELS DE RESUME AUTOMATIQUE DE TEXTES
« SWESUM » & « RESUME »

de Hercules DALIANIS de Aude ACOULON
et Martin HASSEL

Rapport d'activité du 26 septembre 2001
par Aude ACOULON
sous la direction de G.CHOLLET, P.VAILLANT

SOMMAIRE

Evaluation des Catégories 4
Les Dépêches d'agence 4
Les Forums de discussion 5
Les Mails 5
Les Fax 6
Problèmes de « Resume » & « Swesum » 6
Les accents 7
Les fautes d'orthographe 8
Les sigles 9
Les mots d'origine étrangère 9
Les mots de même racine 10
Astuce de « Resume » : Gestion des Connecteurs 11
Conclusion 14

Évaluation des Catégories

Nous tenterons d'évaluer le comportement des deux logiciels sur un même texte, et cela dans quatre catégories. Nous nous limiterons à un **taux de contraction de 30%** pour le résumé.

Les Dépêches d'agence

Dans la majorité des cas, **les deux logiciels ont su redonner de façon partielle un contexte** à chacun des résumés produits :

« Resume » ne supporte pas les chiffres contenant un point (par exemple : 300.000). En effet, « Resume » comporte un module découpage de phrases qui compte une phrase lorsqu'il a trouvé une ponctuation de fin de phrase. « Swesum », lui, ne pose pas ce genre de problème puisqu'il détecte bien un chiffre comportant un point et non deux phrases.

D'autre part, **lorsqu'un texte source ne contient que trois longues phrases, « Swesum » les résume avec une contraction de 75%, car le texte source n'offre pas un assez grand nombre de phrases.** En sortie, le texte résumé est quasiment la copie conforme du texte source. « Resume » ne sélectionnera que les phrases les plus importantes même si celles-ci sont peu nombreuses dans le texte source.

Au point de vue de la longueur des résumés obtenus, nous pouvons dire que **« Swesum » nous indique fréquemment une contraction supérieure à 30 %** pour le résumé produit, alors que « Resume » respecte la longueur de résumé demandé.

Les performances des deux logiciels sont pratiquement identiques. Ainsi, pour un texte long ou court, les deux logiciels ont eu un temps d'exécution similaire :

	Texte1	Texte2	Texte3	Texte4
Resume	5 secondes	6 secondes	10 secondes	5 secondes
Swesum	7 secondes	4 secondes	10 secondes	3 secondes

Du point de vue de la pertinence et de la longueur du résumé produit, « Resume » est plus fiable que « Swesum ». En revanche, **« Swesum » commet moins de fautes de découpage que « Resume »** qui ne gère pas encore assez de paramètres tels que des chiffres contenant des points ou des adresses électroniques contenant des points…Toutefois, **la fiabilité du module de découpage de phrases de « Swesum » dépend de la nature du texte source.**

Les Forums de discussion

En ce qui concerne les deux logiciels, **les fautes d'orthographe ou de grammaire** dans le texte source nuisent au bon repérage des mots-clefs et quelque peu au contexte. De même, **les acronymes ou mots étrangers** (ex : *Newsgroups, From, To, Subject*) ne sont pas « reconnus » par les logiciels.

Les forums de discussion sont pour les deux logiciels, un exercice périlleux en matière de découpage de textes. En effet, « Swesum » et « Resume » sont tous deux, incapables de prendre en compte **les informations essentielles** quand celles-ci sont **séparées, le plus souvent, par des séries d'espaces** (ou tabulations) normées indiquant des informations courtes telles qu'un titre, une date... Ce découpage peut se révéler être gênant quand il s'agit d'une annonce de thèse par exemple.

Étant donné que les deux logiciels fonctionnent sur le fait que la « reconnaissance d'une phrase n'est possible que lorsqu'on rencontre une ponctuation de fin de phrase (? . ; ! ...) » alors il est **impossible de percevoir une phrase écrite sous le coup d'une intonation orale** comme : « Tout d'abord ce n'est pas une affirmation puisque si vous relisez attentivement ma phrase **je dis…je ne pense pas que…mais admettons !** ». En effet, les deux logiciels vont, d'après leur module de découpage de textes, « tronquer » cette phrase en 3 phrases sélectionnées ou non pour le résumé final ! Tout ceci nuit considérablement à la cohérence entre les phrases.

Là aussi, contrairement à « Swesum », le logiciel « Resume » respecte bien la demande initiale de l'utilisateur en réalisant une contraction de 30%.

Lorsque le message d'un internaute sur un forum de discussion, reprend bien la réponse d'un autre interlocuteur faisant référence à une question posée ultérieurement dans le forum, alors le résumé produit par « Swesum » et par « Resume » est excellent puisqu'il redonne en quelques lignes, le sujet et les arguments des deux internautes qui discutent.

En revanche, lorsqu'un **un internaute a volontairement répondu sur un des éléments d'une question posée par un autre interlocuteur** (qui elle, n'est pas incluse dans le résumé)**,** alors le résumé produit par « Swesum » et « Resume » est incohérent car **les phrases sélectionnées évoquent un contexte différent.**

Les Mails

Contrairement à « Swesum », **le logiciel « Resume » supporte une abondance de petits signes stylistiques que l'on peut trouver dans les mails** (tirets rapprochés spécifiant un trait de soulignement, étoiles, ...).

Lorsqu'il y a absence de ponctuation de fin de phrase, « Swesum » a tendance à prendre en compte **tous les mots quels qu'ils soient comme appartenant à une unique phrase**. Le logiciel « Resume », lui, crée un fichier vide.

L'utilisation des connecteurs logiques tels que « donc » et « bref » est très fréquente dans les mails. Le logiciel « Resume » conçu sur le mode de détection de liens

logiques dans un texte peut immédiatement identifier et sélectionner une phrase contenant le connecteur « Donc » par exemple. En revanche, le logiciel « Swesum » qui n'est pas doté de cet outil linguistique, pose plus de problèmes de cohérence entre les phrases que le logiciel « Resume ».

Comme « Swesum », « Resume » garde les chevrons ou séries de chevrons symbolisant la redite ou d'éventuelles réponses aux réponses d'anciens mails, ce qui améliore la lecture du mail.

De la même manière que « Swesum », le logiciel « Resume » nous donne des résumés peu pertinents en ce qui concernent des annonces de thèses, colloques ou réunions d'informations. Ainsi, il manque **des informations essentielles comme le lieu ou la date car celles-ci n'apparaissent qu'une fois et très fréquemment hors phrases**.

Les Fax

Tout comme « Swesum », le logiciel « Resume » **ne sait pas résumer des textes sources peu lisibles**. De ce fait, **le résumé produit ne redonne pas un contexte aussi précis qu'il devrait et demeure très incohérent au niveau syntaxique.** En effet, le logiciel « Resume » comptabilise seulement les mots qu'il « reconnaît » comme mot-clés.

D'une manière générale, le logiciel « Swesum » ne respecte pas le taux de contraction pour le résumé (30 %) contrairement au logiciel « Resume ». Si «Resume » sait gérer plus de cohérence entre les phrases en détectant d'éventuels connecteurs logiques dans les textes sources, en revanche, « Swesum » dispose d'un meilleur module de découpage de textes que « Resume ».

Par ailleurs, nous pouvons dire que les deux logiciels ont un temps d'exécution du programme quasiment identique. Toutefois, « Swesum » et « Resume » posent encore des problèmes d'ordre rédactionnel (repérage de phrases « orales » écrites, détection de signes stylistiques dans les mails, séries d'espaces entre des informations, fautes d'orthographes…).

Problèmes de « Resume » & « Swesum »

Voici la liste des cas possibles dont nous allons nous servir pour évaluer le comportement des deux logiciels face à ces situations de problèmes créées :

- mails non accentués
- mails contenant des mots mal orthographiés

- mails contenant des sigles
- mails contenant des mots d'origine étrangère
- mails contenant des mots de la même racine

Pour éviter que « Resume » n'accorde trop d'importance aux phrases introduites par un connecteur logique, nous avons choisi de construire des textes sans aucun connecteur logique. De la même manière, nous ne tiendrons pas compte de la sémantique et de la cohérence du textes sources.

Les accents

Si etre genereux etait l'apanage du corbeau, alors il faudrait se taire. C'est un animal genereux. Il sait soigner, donner, et garder l'amitie des gens. Le corbeau est altruiste. Fin.

« Resume » :
Il sait soigner, donner, et garder l'amitie des gens.
Le corbeau est altruiste.

« Swesum » :
Si etre genereux etait l'apanage du corbeau, alors il faudrait se taire.
C'est un animal genereux.

Étant donné que le mot « genereux » est présenté sans accents, « Resume » et « Swesum » ne le reconnaissent pas dans le dictionnaire des mot-clés. En effet, « Resume » et « Swesum » vont traiter « genereux » comme un mot et non comme un mot-clef.

Le logiciel « Resume » possède un moteur reposant de la logique, des statistiques. Ainsi, « Resume » sélectionnera une phrase dont il aura calculé un grand poids au niveau des occurrences des mots-clefs qu'elle comporte. Or, d'après ce texte, « Resume » ne compte que 4 mots-clefs dans la phrase : « Si etre genereux etait l'**apanage** du **corbeau**, alors il **faudrait** se **taire**. ». C'est donc la phrase « Il **sait soigner**, **donner**, et **garder** l'amitie des **gens** » qu'il sélectionne pour le résumé car il y compte 5 mots-clefs.

Similairement à « Resume », « Swesum » reconnaît le mot « corbeau » comme mot-clef mais lui non plus ne va pas sélectionner uniquement des phrases qui le comportent. Ainsi, au lieu de sélectionner les deux seules phrases contenant toutes les deux, le mot-clef « corbeau » : « Si etre genereux etait l'apanage du **corbeau**, alors il faudrait se taire. Le **corbeau** est altruiste. », « Swesum » construira son résumé en reprenant deux phrases contenant le mot « genereux » même s'il ne l'a pas identifié comme un mot-clef.

Nous allons tenter de savoir si Swesum désirait bien reprendre le mot « genereux » en entrant le texte suivant :

Etre genereux etait l'apanage du corbeau. Alors il faudrait se taire. C'est un animal genereux. Il sait soigner, donner, et garder l'amitie des gens. Le corbeau est altruiste. Oui il est genereux. Fin.

« Swesum » :
Etre genereux etait l'apanage du corbeau.
Alors il faudrait se taire.
C'est un animal genereux.

« Resume » :
Il sait soigner, donner, et garder l'amitie des gens.
Le corbeau est altruiste.

Là, le comportement de « Swesum » est incohérent car il ne considère véritablement pas le mot « genereux » comme un mot-clef mais bien comme un simple mot, tout comme « Resume ». C'est que les deux logiciels ne savent pas gérer les accents.

Les fautes d'orthographe

Commettons volontairement une faute d'orthographe sur le mot « généreux » que nous transformerons en « généreu ». Décrivons à présent les comportements qui découlent des deux logiciels :

« Swesum » :
Etre généreu était l'apanage du corbeau.
Alors il faudrait se taire.
C'est un animal généreu.

« Resume » :
Il sait soigner, donner, et garder l'amitié des gens.
Le corbeau est altruiste.

Les comportements des deux logiciels sont semblables : ils traitent le mot mal orthographié comme un mot et non comme un mot-clef puisqu'ils ne le reconnaissent pas d'après leurs dictionnaires français.

Les sigles

Maintenant, analysons le comportement des deux logiciels fac aux problèmes que peuvent poser des sigles dans un texte :

L'E.N.S.T forme chaque année des étudiants. C'est un établissement de renom. Plusieurs salles abritent des chercheurs. L'école E.N.S.T est une très bonne école. Il faut décrocher un concours pour y entrer. Fin.

« Swesum » :
L'E.N.S.T forme chaque année des étudiants.
C'est un établissement de renom.
Plusieurs salles abritent des chercheurs.

« Resume » :
T forme chaque année des étudiants.
Plusieurs salles abritent des chercheurs.
L'école E.

« Resume » ne sait pas gérer les sigles car son module de découpage de texte est basé sur le fait qu'une phrase se termine par une ponctuation de fin de phrase. En revanche, « Swesum » sait gérer la présence de sigles car il se base sur le fait qu'une phrase se termine par une ponctuation de fin de phrase suivie d'un espace.

En fait, un sigle ne peut être « reconnu » par un simple logiciel comme « Resume » qui ne sait pas revenir en arrière dans un mot pour déterminer que si « avant un point il existe une lettre alors c'est sûrement un sigle » et qu'il faut savoir que la fin du mot n'est pas « E.N**.S** » mais bien « E.N.S**.T** ». Par conséquent « Swesum » sait reconnaître la fin d'un sigle .

Cependant, d'après l'évaluation faite uniquement sur le logiciel « Swesum », il a été démontré que « Swesum » ne reconnaissait pas l**es acronymes ou les sigles**, **même si l'objet du mail en dépend** (ex : une annonce d'emploi décrivant *un poste en **CDI** ou en **CDD** demandant lettre et **CV***). Nous pouvons donc penser que la fiabilité du module de découpage de textes du logiciel « Swesum » est difficilement stable ou logique.

Les mots d'origine étrangère

A présent, intéressons-nous aux mots d'origine étrangère ou néologismes.

Microsoft veut détenir le monopole.
Windows est son bébé qui a tout déclenché.
Un operating system ingénieux.
Windows épate tout le monde.
Windows n'est pour eux, qu'un logiciel graphique.
Sauf les Linuxiens.
Fin.

« Swesum » :
Microsoft veut détenir le monopole.
Windows est son bébé qui a tout déclenché.
Un operating system ingénieux.

« Resume » :
Microsoft veut détenir le monopole.
Windows est son bébé qui a tout déclenché.

Ici, les deux logiciels ont adopté un comportement différent. Tout d'abord, du point de vue de la longueur, « Swesum » a retenu trois phrases alors que « Resume » n'en a gardé que deux. Il semble que le mot « Windows » ait été reconnu par les deux logiciels comme un mot-clef. De même pour le mot « Microsoft » repris les deux logiciels alors que celui-ci n'a qu'une occurrence dans le texte source.

En fait, pour « Resume », les phrases « Microsoft **veut détenir** le **monopole** » et « Windows est son **bébé** qui a tout **déclenché**. » contiennent respectivement 3 et 2 mot-clés. De son côté, « Resume » a une attitude moins claire puisqu'il prend directement les trois premières phrases du texte source.

Les mots de même racine

L'enfantement, douloureuse étape pour toutes.
Les adultes savent mesurer et doser leurs choix.
L'enfance est un rêve qui nous préoccupe tous.
Les jeunes ne savent pas faire de demi-mesure.
Ils veulent tout blanc, tout noir.
Tout le monde a été enfant.
C'est de là que vienne leur innocence de pensée.
Fin.

« Swesum » :
L'enfantement, douloureuse étape pour toutes.
Les adultes savent mesurer et doser leurs choix.
Les jeunes ne savent pas faire de demi-mesure.

« Resume » :
Les adultes savent mesurer et doser leurs choix.
Les jeunes ne savent pas faire de demi-mesure.

Tout le monde a été enfant.

Le logiciel « Swesum » ne sait pas gérer les mots appartenant à la même racine. Son comportement est difficilement rationnel puisqu'il n'a trouvé aucun mot-clef ayant une deuxième occurrence dans le texte source.
De même pour « Resume », les mots de même étymologie n'ont pas de valeur. Pour construire un résumé, celui-ci va se baser sur des statistiques mathématiques

Il semble difficile pour les deux logiciels de savoir gérer des éléments « inexistants » tels que des mots mal accentués ou mal orthographiés. De la même façon, les deux logiciels ne présentent pas une pertinence linguistique aussi poussée que celle de reconnaître des mots de même racine et donc d'en déduire qu'ils sont tous des mots-clés.

Toutefois, le logiciel « Resume » est le seul des deux logiciels qui contient un apport linguistique indéniable : la gestion des liens logiques dans un texte.

Astuce de « Resume » : Gestion des Connecteurs

Il y a peu de différence entre les deux logiciels du point de vue de l'aptitude à produire un résumé et du point de vue des performances. Toutefois, nous allons tenter de découvrir les comportements des deux logiciels sur un même texte source.

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Bref, le corbeau est un animal généreux. Finalement, il sait soigner, donner et garder l'amitié des gens. Le corbeau est un animal altruiste. Fin.

« Resume » :
Bref, le corbeau est un animal généreux.
Finalement, il sait soigner, donner et garder l'amitié des gens

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Bref, le corbeau est un animal généreux.

Ici, nous pouvons remarquer d'emblée, que les deux logiciels ont produit un résumé composé de deux phrases. Le premier logiciel, « Resume », a privilégié les deux seules phrases du texte introduites par un connecteur logique.

Le deuxième logiciel, « Swesum », a gardé deux phrases contenant les mots-clefs « corbeau » et « animal ». Cependant, d'après le texte source, si nous suivons la logique de

fonctionnement du logiciel, nous aurions dû obtenir un résumé fait de phrases comportant à la fois le mot-clef « corbeau » et le mot-clef « animal » : « Bref, le **corbeau** est un **animal** généreux. » et « Le **corbeau** est un **animal** altruiste. »

Or, le logiciel « Swesum » n'a gardé qu'une seule phrase comportant en même temps les deux mots-clés « corbeau » et « animal » et une phrase introduite par le connecteur logique « Bref ».

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Le corbeau est un animal généreux. Il sait soigner, donner et garder l'amitié des gens. Finalement, le corbeau est un animal altruiste. Fin.

« Resume » :
Le corbeau est un animal généreux.
Finalement, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

« Resume » a pris la seule phrase introduite par un connecteur logique et une phrase comportant les mots « corbeau » et « animal » qu'il a détecté comme étant des mots-clés. « Swesum », lui, a repris les deux phrases qu'il avait obtenu comme pour le précédent résumé. Par conséquent, nous pouvons penser que « Swesum » en ne donnant pas de valeur aux phrases introduites par des connecteurs logiques, ne témoigne pas d'une réelle pertinence linguistique.

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Le corbeau est un animal généreux. Il sait soigner, donner et garder l'amitié des gens. Bref, le corbeau est un animal altruiste. Fin.

« Resume » :
Le corbeau est un animal généreux.
Bref, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

Nous pouvons confirmer à travers ces deux résumés que le logiciel « Swesum » n'a pas changé de comportement en détectant une phrase introduite par le connecteur logique « Bref ».

Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.
Il sait soigner, donner et garder l'amitié des gens.
Donc, le corbeau est un animal altruiste.
Fin.

« Resume » :
Le corbeau est un animal généreux.
Donc, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

Ici encore, « Swesum » n'a pas changé de logique de comportement face à une phrase introduite par le connecteur logique « Donc » qui illustrait pourtant, d'après notre compétence linguistique, de manière très forte, un résultat.

En conclusion, nous avons pu remarquer que les deux logiciels respectaient bien le taux de contraction pour le résumé (30 %), car ils ne retenaient tous deux que deux phrases. Par ailleurs, nous pouvons dire que le logiciel « Resume » est plus pertinent que le logiciel « Swesum ». En effet, « Resume » sait gérer la présence de connecteurs logiques qui sont des mots-clés servant à partitionner un texte long en paragraphes par exemple.

Conclusion

Les deux logiciels présentent un comportement semblable en ce qui concerne la gestion des accents, des fautes d'orthographe et des mots de même étymologie. Ainsi, « Swesum » et « Resume » ne savent pas reconnaître des mots qu'ils soient non accentués ou mal orthographiés comme des mot-clés. De même pour les mots de même étymologie, les deux logiciels ne savent pas distinguer une même racine et donc un lien de familiarité entre des mots de même racine pouvant exprimer quelque chose de semblable.

« Swesum » ne respecte jamais le taux de contraction de textes demandé au départ, par l'utilisateur. En effet, le résumé produit, dépasse fréquemment les 30% du texte initial. « Resume » est un logiciel qui présente une assez bonne pertinence au point de vue linguistique puisqu'il sait gérer les phrases introduites par des connecteurs logiques. Le résumé obtenu apparaît logique car structuré.

Pourtant, les deux logiciels présentent un module de statistiques d'occurrences de mots-clés fragile. Tout dépend de la nature du texte source, de la longueur des phrases et de la place de ces phrases dans le texte source.

En effet, du point de vue logique, le fait même de donner **une valeur à des mots ou expressions** (comme le ferait un cerveau humain lors de la lecture d'un journal afin de décoder un cheminement de logique de pensée), est un atout pertinent pour rassembler des phrases-clés disséminées **de manière logique** dans tout le texte.

COMPARAISON
DES

LOGICIELS DE RESUME AUTOMATIQUE DE TEXTES
« SWESUM » & « RESUME »

de Hercules DALIANIS de Aude ACOULON
et Martin HASSEL

Rapport d'activité du 26 septembre 2001
par Aude ACOULON
sous la direction de G.CHOLLET, P.VAILLANT

SOMMAIRE

Astuce de « Resume » : Gestion des Connecteurs....16
Problèmes de « Resume » & « Swesum »....18
Les accents....19
Les fautes d'orthographe....20
Les sigles....20
Les mots d'origine étrangère....21
Les mots de même racine....22
Conclusion....23

Astuce de « Resume » : Gestion des Connecteurs

Il y a peu de différence entre les deux logiciels du point de vue de l'aptitude à produire un résumé et du point de vue des performances. Toutefois, nous allons tenter de découvrir les comportements des deux logiciels sur un même texte source.

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Bref, le corbeau est un animal généreux. Finalement, il sait soigner, donner et garder l'amitié des gens. Le corbeau est un animal altruiste. Fin.

« Resume » :
Bref, le corbeau est un animal généreux.
Finalement, il sait soigner, donner et garder l'amitié des gens

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Bref, le corbeau est un animal généreux.

Ici, nous pouvons remarquer d'emblée, que les deux logiciels ont produit un résumé composé de deux phrases. Le premier logiciel, « Resume », a privilégié les deux seules phrases du texte introduites par un connecteur logique.

Le deuxième logiciel, « Swesum », a gardé deux phrases contenant les mots-clefs « corbeau » et « animal ». Cependant, d'après le texte source, si nous suivons la logique de fonctionnement du logiciel, nous aurions dû obtenir un résumé fait de phrases comportant à la fois le mot-clef « corbeau » et le mot-clef « animal » : « Bref, le **corbeau** est un **animal** généreux. » et « Le **corbeau** est un **animal** altruiste. »

Or, le logiciel « Swesum » n'a gardé qu'une seule phrase comportant en même temps les deux mots-clés « corbeau » et « animal » et une phrase introduite par le connecteur logique « Bref ».

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Le corbeau est un animal généreux. Il sait soigner, donner et garder l'amitié des gens. Finalement, le corbeau est un animal altruiste. Fin.

« Resume » :
Le corbeau est un animal généreux.
Finalement, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

« Resume » a pris la seule phrase introduite par un connecteur logique et une phrase comportant les mots « corbeau » et « animal » qu'il a détecté comme étant des mots-clés. « Swesum », lui, a repris les deux phrases qu'il avait obtenu comme pour le précédent résumé. Par conséquent, nous pouvons penser que « Swesum » en ne donnant pas de valeur aux phrases introduites par des connecteurs logiques, ne témoigne pas d'une réelle pertinence linguistique.

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Le corbeau est un animal généreux. Il sait soigner, donner et garder l'amitié des gens. Bref, le corbeau est un animal altruiste. Fin.

« Resume » :
Le corbeau est un animal généreux.
Bref, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

Nous pouvons confirmer à travers ces deux résumés que le logiciel « Swesum » n'a pas changé de comportement en détectant une phrase introduite par le connecteur logique « Bref ».

Si mentir était l'apanage du corbeau, alors il faudrait se taire. Le corbeau est un animal généreux. Il sait soigner, donner et garder l'amitié des gens. Donc, le corbeau est un animal altruiste. Fin.

« Resume » :
Le corbeau est un animal généreux.
Donc, le corbeau est un animal altruiste.

« Swesum » :
Si mentir était l'apanage du corbeau, alors il faudrait se taire.
Le corbeau est un animal généreux.

Ici encore, « Swesum » n'a pas changé de logique de comportement face à une phrase introduite par le connecteur logique « Donc » qui illustrait pourtant, d'après notre compétence linguistique, de manière très forte, un résultat.

En conclusion, nous avons pu remarquer que les deux logiciels respectaient bien le taux de contraction pour le résumé (30 %), car ils ne retenaient tous deux que deux phrases. Par ailleurs, nous pouvons dire que le logiciel « resume » est plus pertinent que le logiciel « Swesum ». En effet, « Resume » sait gérer la présence de connecteurs logiques qui sont des mots-clés servant à partitionner un texte long en paragraphes par exemple.

Problèmes de « Resume » & « Swesum »

Voici la liste des cas possibles qui ne pourront être traités de manière optimale par le logiciel :

- mails non accentués
- mails contenant des mots mal orthographiés

- mails contenant des sigles
- mails contenant des mots d'origine étrangère
- mails contenant des mots de la même racine

Les accents

Pour éviter que « Resume » n'accorde trop d'importance aux phrases introduites par un connecteur logique, nous avons choisi de construire un texte sans aucun connecteur logique. De la même manière, nous ne tiendrons pas compte de la sémantique et de la cohérence du texte source.

Si etre genereux etait l'apanage du corbeau, alors il faudrait se taire. C'est un animal genereux. Il sait soigner, donner, et garder l'amitie des gens. Le corbeau est altruiste. Fin.

« Resume » :
Il sait soigner, donner, et garder l'amitie des gens.
Le corbeau est altruiste.

« Swesum » :
Si etre genereux etait l'apanage du corbeau, alors il faudrait se taire.
C'est un animal genereux.

Étant donné que le mot « genereux » est présenté sans accents, « Resume » et « Swesum » ne le reconnaissent pas dans le dictionnaire des mot-clés. En effet, « Resume » et « Swesum » vont traiter « genereux » comme un mot et non comme un mot-clef.

Le logiciel « Resume » possède un moteur reposant de la logique, des statistiques. En effet, « Resume » sélectionnera une phrase dont il aura calculé un grand poids au niveau des occurrences des mots-clefs qu'elle comporte. Or, d'après ce texte, « Resume » ne compte que 4 mots-clefs dans la phrase : « Si etre genereux etait l'**apanage** du **corbeau**, alors il **faudrait** se **taire**. ». C'est donc la phrase « Il **sait soigner**, **donner**, et **garder** l'amitie des **gens** » qu'il sélectionne pour le résumé car il y compte 5 mots-clefs.

Similairement à « Resume », « Swesum » reconnaît le mot « corbeau » comme mot-clef mais lui non plus ne va pas sélectionner uniquement des phrases qui le comportent. Ainsi, au lieu de sélectionner les deux seules phrases contenant toutes les deux, le mot-clef « corbeau » : « Si etre genereux etait l'apanage du **corbeau**, alors il faudrait se taire. Le **corbeau** est altruiste. », « Swesum » construira son résumé en reprenant deux phrases contenant le mot « genereux » même s'il ne l'a pas identifié comme un mot-clef.

Nous allons tenter de savoir si Swesum désirait bien reprendre le mot « genereux » en entrant le texte suivant :

Etre genereux etait l’apanage du corbeau.
Alors il faudrait se taire.
C’est un animal genereux.
Il sait soigner, donner, et garder l’amitie des gens.
Le corbeau est altruiste.
Oui il est genereux.
Fin.

« Swesum » :
Etre genereux etait l’apanage du corbeau.
Alors il faudrait se taire.
C’est un animal genereux.

« Resume » :
Il sait soigner, donner, et garder l’amitie des gens.
Le corbeau est altruiste.

Là, le comportement de « Swesum » est incohérent car il ne considère véritablement pas le mot « genereux » comme un mot-clef mais bien comme un simple mot, tout comme « Resume ». C’est que les deux logiciels ne savent pas gérer les accents.

Les fautes d’orthographe

Commettons volontairement une faute d’orthographe sur le mot « généreux » que nous transformerons en généreu. Décrivons à présent les comportements :

« Swesum » :
Etre généreu était l’apanage du corbeau.
Alors il faudrait se taire.
C’est un animal généreu.

« Resume » :
Il sait soigner, donner, et garder l’amitié des gens.
Le corbeau est altruiste.

Les comportements des deux logiciels sont semblables : ils traitent le mot mal orthographié comme un mot et non comme un mot-clef puisqu’ils ne le reconnaissent pas d’après leurs dictionnaires français.

Les sigles

Maintenant, analysons le comportement des deux logiciels fac aux problèmes que peuvent poser des sigles dans un texte :

L’E.N.S.T forme chaque année des étudiants.
C’est un établissement de renom.
Plusieurs salles abritent des chercheurs.
L’école E.N.S.T est une très bonne école.
Il faut décrocher un concours pour y entrer.
Fin.

« Swesum » :
L’E.N.S.T forme chaque année des étudiants.
C’est un établissement de renom.
Plusieurs salles abritent des chercheurs.

« Resume » :
T forme chaque année des étudiants.
Plusieurs salles abritent des chercheurs.
L’école E.

« Resume » ne sait pas gérer les sigles car son module de découpage de texte est basé sur le fait qu’une phrase se termine par une ponctuation de fin de phrase. En revanche, « Swesum » sait gérer la présence de sigles car il se base sur le fait qu’une phrase se termine par une ponctuation de fin de phrase suivie d’un espace.

En fait, un sigle ne peut être « reconnu » par un simple logiciel comme « Resume » qui ne sait pas revenir en arrière dans un mot pour déterminer que si « avant un point il existe une lettre alors c’est sûrement un sigle » et qu’il faut savoir que la fin du mot n’est pas « E.N**.S** » mais bien « E.N.S**.T** ». Par conséquent « Swesum » sait reconnaître la fin d’un sigle .

Les mots d’origine étrangère

A présent, intéressons-nous aux mots d’origine étrangère ou néologismes.

Microsoft veut détenir le monopole.
Windows est son bébé qui a tout déclenché.
Un operating system ingénieux.
Windows épate tout le monde.
Windows n’est pour eux, qu’un logiciel graphique.
Sauf les Linuxiens.
Fin.

« Swesum » :
Microsoft veut détenir le monopole.
Windows est son bébé qui a tout déclenché.
Un operating system ingénieux.

« Resume » :
Microsoft veut détenir le monopole.

Windows est son bébé qui a tout déclenché.

Ici, les deux logiciels ont adopté un comportement différent. Tout d'abord, du point de vue de la longueur, « Swesum » a retenu trois phrases alors que « Resume » n'en a gardé que deux. Il semble que le mot « Windows » ait été reconnu par les deux logiciels comme un mot-clef. De même pour le mot « Microsoft » repris les deux logiciels alors que celui-ci n'a qu'une occurrence dans le texte source.

En fait, pour « Resume », les phrases « Microsoft **veut détenir** le **monopole** » et « Windows est son **bébé** qui a tout **déclenché**. » contiennent respectivement 3 et 2 mot-clés.
De son côté, « Resume » a une attitude moins claire puisqu'il prend directement les trois premières phrases du texte source.

Les mots de même racine

L'enfantement, douloureuse étape pour toutes.
Les adultes savent mesurer et doser leurs choix.
L'enfance est un rêve qui nous préoccupe tous.
Les jeunes ne savent pas faire de demi-mesure.
Ils veulent tout blanc, tout noir.
Tout le monde a été enfant.
C'est de là que vienne leur innocence de pensée.
Fin.

« Swesum » :
L'enfantement, douloureuse étape pour toutes.
Les adultes savent mesurer et doser leurs choix.
Les jeunes ne savent pas faire de demi-mesure.

« Resume » :
Les adultes savent mesurer et doser leurs choix.
Les jeunes ne savent pas faire de demi-mesure.
Tout le monde a été enfant.

Le logiciel « Swesum » ne sait pas gérer les mots appartenant à la même racine. Son comportement est difficilement rationnel puisqu'il n'a trouvé aucun mot-clef ayant une deuxième occurrence dans le texte source.
De même pour « Resume », les mots de même étymologie n'ont pas de valeur. Pour construire un résumé, celui-ci va se baser sur des statistiques mathématiques

Il semble difficile pour les deux logiciels de savoir gérer des éléments « inexistants » tels que des mots mal accentués ou mal orthographiés. Cependant, « Swesum » est capable lui, de respecter le découpage d'une phrase jusqu'à sa ponctuation de fin de phrase et cela même quand il s'agit d'un sigle. De la même façon, les deux logiciels ne présentent pas une

pertinence linguistique aussi poussée que celle de reconnaître des mots de même racine et donc d'en déduire qu'ils sont tous des mots-clés.

Conclusion

Les deux logiciels présentent un comportement semblable en ce qui concerne la gestion des accents, des fautes d'orthographe et des mots de même étymologie. Ainsi, « Swesum » et « Resume » ne savent pas reconnaître des mots qu'ils soient non accentués ou mal orthographiés comme des mot-clés. De même pour les mots de même étymologie, les deux logiciels ne savent pas distinguer une même racine et donc un line de familiarité entre des mots de même racine pouvant exprimer quelque chose de semblable.

« Swesum » est un logiciel qui présente un assez bon module de découpage de textes pour les sigles. En effet, « swesum » gère la présence des points entre les lettres d'un sigle. Le sigle est découpé en son entier et est considéré comme un mot. « Resume » peut reconnaître

un sigle si on enlève ses points et qu'il soit contenu dans un des dictionnaires de mots-clés (ex : ENST et non plus E.N.S.T).

« Resume » est un logiciel qui présente une assez bonne pertinence au point de vue linguistique puisqu'il sait gérer les phrases introduites par des connecteurs logiques. Le résumé obtenu apparaît logique car structuré.

Pourtant, les deux logiciels présentent un module de statistiques d'occurrences de mots-clés fragile. Tout dépend de la nature du texte source, de la longueur des phrases et de la place des phrases dans le texte source.

En effet, du point de vue logique, le fait même de donner une valeur à des mots ou expressions (comme le ferait un cerveau humain lors de la lecture d'un journal afin de décoder un cheminement de logique de pensée), est un atout pertinent pour rassembler des phrases-clés disséminées de manière logique dans tout le texte.